TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra softwarového inženýrství

DIPLOMOVÁ PRÁCE

Softwarové vybavení CAN uzlu

Software equipment CAN node

Liberec 2004 Vladimír Harbich

Anotace:

Cílem diplomové práce bylo vytvořit návrh aplikačního protokolu sběrnice CAN. Tento navržený protokol je implementován do komunikačního uzlu pro sběrnici CAN, který je na bázi mikrořadiče AT89C52 s CAN řadičem 82527. Dále je implementován do mikropočítačových systémů s mikrořadičem Infineon C167CR.

Aplikační protokol je navržen s ohledem na jednoduchost a tak, aby nekladl velké hardwarové nároky. Komunikace mezi periferiemi probíhá pevně stanovenými způsoby v předdefinovaných rámcích Obsah jednotlivých rámců i způsob přenosu lze konfigurovat.

Řídicí programy pro CAN uzel na bázi x52 i moduly s mikrořadičem C167CR jsou vytvořeny v jazyce C ve vývojovém prostředí Keil µVision2.

Pro komunikační modul s mikrořadičem AT89C52 je v jazyce Borland Delphi 5 napsána konfigurační aplikace. Ta umožňuje konfigurovat parametry uzlu z PC přes standardní sériový port.

Abstract:

The aim of this diploma thesis was to create an application protocol proposal of busbar CAN. Proposed protocol is implemented into the communication node for CAN's bus that works on the AT89C52 microcontroller base with 82527 CAN controller. Protocol is then implemented into the microcomputer's systems with Infineon C167CR microcontroller.

Application protocol is proposed in reference to the simplicity and with minimum hardware requirements. Communication between peripheries is achieved by fixed ways with templates. It is possible to configure the capacity and the transfer's way of particular frames.

Control programs for CAN node on the x52 base and modules with C167CR microcontroller are created in C language within Keil µVision2 development environment.

Communication application is written for communication module with AT89C52 microcontroller with using Borland Delphi 5 language. This provides possibility to configure node parameters from PC via the standard serial port.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 21.5.2004

Podpis:

Obsah:

1 Předmluva

Na tomto místě bych rád poděkoval Ing. Josefu Grosmanovi za cenné rady při konzultacích a během tvorby této diplomové této práce.

Dále děkuji své rodině a přátelům za podporu během studia.

2 Úvod

Sériové komunikační sběrnice pro přenos dat v řídicích a měřicích systémech již od svého vzniku procházejí neustálým vývojem. Požadavky současné doby kladou velmi vysoké nároky nejen na zajištění datového toku z hlediska spolehlivosti, rychlosti přenosu a kapacity sběrnice, ale i na softwarové služby spojené se sériovou komunikací. Právě toto softwarové vybavení je jedním z důležitých kritérií při rozhodování o použití sběrnice pro konkrétní aplikaci.

Implementaci komunikačních systémů v praxi lze rozdělit do dvou základních skupin. První skupinu můžeme chápat jako produkty vyráběné ve velkých sériích (např. obráběcí stroje, dopravní prostředky atd.). Tyto produkty jsou zpravidla vybavovány sběrnicemi, jejichž software je na nejnižší úrovni, který vyhovuje pouze dané aplikaci. Druhou skupinou jsou malosériové celky (např. komponenty pro distribuované měřicí nebo řídicí systémy, automatizaci budov apod.). U této skupiny narozdíl od první je kladen velký důraz na flexibilitu zařízení. To je důvod, proč jsou tyto systémy vybavovány sběrnicemi se softwarem na vyšší úrovni, který zajišťuje flexibilitu a kompatibilitu zařízení od různých výrobců.

Cílem této diplomové práce je vytvořit návrh jednoduchého protokolu na vyšší aplikační úrovni pro sběrnici CAN (Controller Area Network) pro účely mikropočítačové laboratoře, jeho implementaci a ověření funkčnosti.

Aplikační protokol má být vyvinut tak, aby bylo možné jej implementovat do komunikačního uzlu na bázi mikrokontroleru Atmel AT89C52 a výukových modulů C167CR/CAN Evaluation And Training Kit s mikropočítačem Infineon C167CR.

3 Stručný popis sběrnice CAN

3.1 Historie a současnost CAN

Sériový komunikační protokol CAN (Controller Area Network) byl oficiálně uveden v roce 1986 německou firmou Robert Bosch GmbH. Byl vyvinut pro použití v automobilovém průmyslu ke komunikaci mezi řídicími jednotkami, snímači a akčními členy. Z tohoto důvodu byl kladen důraz na velmi vysoké zabezpečení přenosu a odolnost vůči okolním rušivým vlivům. Díky robustnosti se tento protokol začal uplatňovat i v jiných oblastech než v automobilovém průmyslu.

I v dnešní době bezmála dvacet let po uvedení je tento komunikační protokol jedním z nejpoužívanějších protokolů nejen v automobilovém průmyslu, ale i v jiných odvětvích, jako je například řízení v reálném čase, automatizace strojů, budov apod. Stále je velkým konkurentem jiným komunikačním protokolům a je hojně nasazován i do netypických řešení.

Protokol CAN je standardizován mezinárodní normou ISO 11898.

3.2 Začlenění do referenčního modelu ISO/OSI

Obecně platí, že vývoj jakéhokoliv nového komunikačního systému je relativně složitý proces, u kterého je vhodné provést dekompozici. Proto bylo vytvořeno standardizované doporučení zásad ISO 7498 Open Systems Interconnect (nazývané též referenční model ISO/OSI). Toto doporučení obsahuje definici tzv. komunikačního protokolu. Protokoly jsou členěny do sedmi vrstev, které jsou znázorněné v tab. 1. Platí zásada, že mezi sebou mohou komunikovat pouze vrstvy, které jsou na stejné úrovni. V komunikačních protokolech typu „fieldbus“ (do kterého patří i CAN) ovšem není třeba využívat všech služeb jednotlivých vrstev, proto se tento model běžně redukuje a jsou ponechány služby pouze fyzické, linkové a aplikační vrstvy.

název vrstvy	příklady funkce vrstvy
Aplikační	*rozhraní k aplikacím* *přenos souborů* *vzdálený přístup*
Prezentační	*reprezentace dat* *kódování* *komprimace*
Relační	*řízení spojení* *navazování a ukončování* *synchronizační body*
Transportní	*přizpůsobovací vrstva* *správa spojení* *rozlišení aplikací*
Síťová	*směrování* *vyvažování zátěže* *řízení sítě*
Linková	*přístup k médiu* *detekce chyb* *definice rámce*
Fyzická	*přenos bitů* *mechanické a elektrické vlastnosti* *kabely, konektory, napětí...*

tab. 1: Model ISO/OSI

Protokoly fyzické vrstvy obsahují definice vlastního přenosového média, specifikace fyzikálních veličin pro přenos informace a definice topologie sítě. Součástí je též zajištění synchronizace vysílače a přijímače.

Linková vrstva obvykle zahrnuje dvě základní služby. První službou je řešení přístupu jednotlivých uzlů k přenosovému médiu (angl. Medium Access Control, MAC) a druhou službou je zajištění vlastního přenosu dat (angl. Logical Link Control, LLC).

CAN přenáší data nejčastěji po dvouvodičové diferenciální sběrnici s charakteristickou impedancí 120 Ω. Maximální rychlost datového toku je 1 Mb/s při délce sběrnice 40 m. Firma Bosch vydala specifikaci obsahující definice fyzického a linkového protokolu CAN. V současné době jsou aktuální dvě verze označované jako CAN 2.0A (standardní formát rámce s 11-bitovým, identifikátorem) a CAN 2.0B (rozšířený formát rámce s 29-bitovým identifikátorem). V těchto specifikacích jsou definovány čtyři možné typy přenášených rámců: datový, žádost o data, chybový a rámec přetížení. Protokol CAN využívá kolizní přístupovou metodu ke sběrnici

označovanou jako CSMA/CD+AMP (Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority).

Aplikační protokol definuje pravidla komunikace na úrovni uživatelské aplikace. Dává význam přenášeným datům (např. hodnoty od senzorů, příkazy akčním členům apod.). Obvykle bývá založen na obecném objektovém modelu komunikace mezi zařízeními. Tato vrstva referenčního modelu protokolu CAN není standardizována normou ISO 11898, ani není obsažena ve specifikacích od firmy Bosch, proto na této úrovni existuje mnoho nekompatibilních protokolů.

3.3 Organizace CiA

Sdružení CiA (CAN in Automation) je nezisková organizace výrobců a uživatelů komponent sběrnice CAN. Vznikla před více než deseti lety a díky velkému úsilí ve formě vydávání standardů má značný podíl na tom, že CAN je jednou z nejrozšířenějších sběrnic. V současnosti organizuje školící akce, vzdělávací kurzy, mezinárodní konference, vydává publikace a standardy, která se týkají protokolu CAN.

Dále podporuje vývoj produktů vyšších aplikačních vrstev (např. CAL-CAN Aplication Layer). V současnosti tato organizace podporuje tři protokoly vyšší aplikační vrstvy: CANopen , CAN Kingdom a DeviceNet.

4 Návrh aplikačního protokolu

Při návrhu byl brán zřetel na jednoduchost z důvodu omezených hardwarových možnosti komunikačního modulu s mikrokontrolerem AT89C52. Jedním z těchto omezení je datová paměť s kapacitou pouhých 256 bajtů. Předem je třeba upozornit, že tento protokol není kompatibilní s žádným dostupným protokolem.

4.1 Základní vlastnosti

Navrhovaný aplikační protokol umožňuje připojení na sběrnici CAN až 128 zařízení, z nichž jedno realizuje funkci řídicí jednotky (Master) a ostatní jednotky jsou podřízené (Slave). Přenos všech dat probíhá v rámcích se standardním formátem s 11-bitovým identifikátorem podle specifikace CAN 2.0A. Přenosová rychlost a vzorkovací bod jsou pevně nastavené hodnoty v každém zařízení. **Přenosová rychlost je 100 kb/s při vzorkovacím bodě 75 %.**

Je umožněno vysílat nebo přijímat předdefinované rámce, které jsou určeny pro komunikaci mezi periferiemi všech zařízení na sběrnici. Tyto datové rámce lze přenášet pevně stanovenými způsoby, které je umožněno konfigurovat podle potřeb uživatele. Dále tento navrhovaný aplikační protokol uživateli umožňuje stanovit, jaká data budou přenášena v datové oblasti těchto rámců. Veškerou konfiguraci je umožněno provádět pomocí komunikace po sběrnici CAN, popřípadě po jiném rozhraní, kterým je dané zařízení vybaveno.

Jsou pevně definovány různé stavy, ve kterých může být zařízení provozováno. Stav každého zařízení lze řídit po sběrnici CAN ze vzdálené jednotky, pomocí předdefinovaných rámců.

K synchronizaci všech zařízení na sběrnici je definován speciální synchronizační rámec, který každých 100 ms vysílá zařízení „Master“ okolním jednotkám.

4.2 Identifikace zařízení v síti

Každé ze zařízení s navrhovaným aplikačním protokolem, které má být připojené ke sběrnici CAN, musí být možné identifikovat. K tomuto účelu byl navržen **identifikátor modulu**, kterým je sedmibitové číslo (0 až 127). Tento identifikátor je

pevně programově nastaven v každém zařízení a není možné jej modifikovat. Pomocí identifikátoru modulu se stanovují identifikátory rámců pro konfigurační účely. Dále je požíván pro příkazy určené k řízení stavů zařízení.

4.3 Objektový slovník

Nedílnou součástí každé jednotky je tzv. **objektový slovník**, je to tabulka jejíž záznamy tvoří předdefinované objekty. Tyto objekty mají rozhodující vliv na přenos dat po sběrnici CAN. Položky této tabulky jsou přístupné přes rozhraní CAN. Každý záznam objektového slovníku je zpřístupněn prostřednictvím 16-bitového indexu a 8-bitového subindexu.

Přehled předdefinovaných objektů:

- Mapovací parametry.
- Komunikační parametry.
- Vnitřní události.
- Fyzické vstupy a výstupy (Aplikační objekty).

Index	Obsah
0000 - 00FF	Rezervováno
0100 - 02FF	Mapovací parametry
0300 - 04FF	Komunikační parametry
0500 - 05FF	Vnitřní události
0600 - 0FFF	Aplikační objekty
1000 - FFFF	Rezervováno

tab. 2: Umístění objektů v objektovém slovníku

4.3.1 Mapovací a komunikační parametry

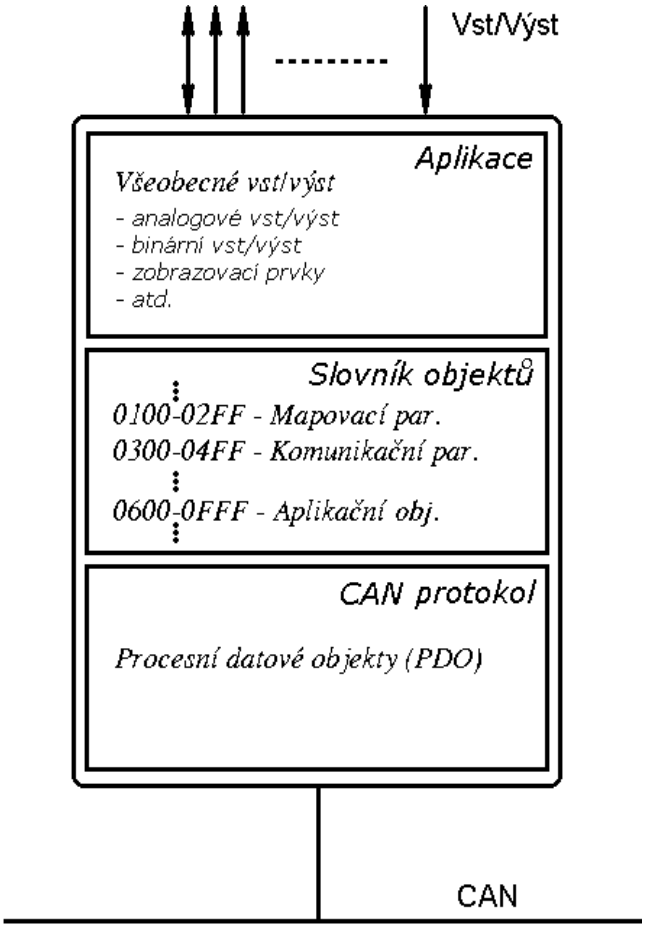
Záznamy mapovacích parametrů na indexech 0100 až 02FF hex jsou ke čtení i k zápisu, umožňují tzv. mapování procesních datových objektů. Komunikační parametry procesních datových objektů na indexech 0300 až 04FF hex jsou rovněž pro čtení i zápis. Popis, účel a používání mapovacích a komunikačních parametrů je detailně popsáno v kap. 4.4.2 a 4.4.3.

4.3.2 Vnitřní události

Umožňuje-li zařízení reagovat na nějakou vnitřní nebo vnější událost (např. vypršení časového limitu, stisk tlačítka apod.), lze této události využít pro vyslání datového rámce (je to jeden ze způsobů přenosu procesních dat, který je popsán v kap. 4.4.4). Podmínkou je, aby této události byl ve slovníku objektů přiřazen index z vyhrazeného rozsahu 0500 až 05FF hex a aby v softwarové obsluze této události bylo umožněno vyslání procesního datového objektu, pokud má v komunikačních parametrech nastaven přenos vyvolaný právě touto událostí.

4.3.3 Aplikační objekty

Pomocí aplikačních objektů v objektovém slovníku je vytvořeno rozhraní mezi sběrnicí CAN a uživatelskými vstupy a výstupy komunikačního uzlu. Za aplikační objekty lze považovat všechny vstupy a výstupy modulu, které mají být přenášeny po sběrnici. Těmto objektům jsou vyhrazeny indexy v rozmezí 0600 až 0FFF hex.



obr. 1: Model zařízení

Jako příklad aplikačního objektu lze uvést signál z binárního vstupu, který má být přenášen po sběrnici. Při programování komunikačního modulu se tomuto vstupu přiřadí některý z vyhrazených indexů objektového slovníku. Při více binárních vstupech lze s výhodou užít systém subindexace, což si lze představit jako jednotku s více binárními vstupy, kde je každý vstup uložen na shodném indexu, ale odlišných subindexech (0 až 255).

Každé naprogramované vstupně-výstupní zařízení potom obsahuje určitý počet aplikačních objektů umístěných na příslušných indexech. Tyto vazby mezi periferiemi a indexy je nutné znát při mapování aplikačních objektů do procesních datových objektů (viz kap. 4.4.2).

Každý **aplikační objekt** v navrhovaném aplikačním protokolu může obsahovat data **o délce od jednoho do osmi bajtů**.

4.4 Komunikační model

Komunikační model popisuje používané komunikační objekty, poskytované služby v síti a principy přenosu jednotlivých zpráv. V komunikačním modelu jsou definovány čtyři typy komunikačních objektů:

- Procesní datové objekty (PDO).
- Servisní datové objekty (SDO).
- Objekty spojené se správou sítě (NMT).
- Speciální synchronizační objekt (SYNC).

4.4.1 Procesní datové objekty (PDO)

Procesní data lze chápat jako vstupní nebo výstupní data (aplikační objekty) jednotlivých modulů, které jsou dostupná uživateli. Příkladem mohou být snímané hodnoty napětí AD převodníku, kód stisknuté klávesy, příkaz k rozsvícení LED apod. Aby tato vstupně-výstupní data mohla být přenášena uvnitř procesních datových objektů, musí být definována ve slovníku objektů jako tzv. aplikační objekty.

Přenos procesních datových objektů (PDO) je nepotvrzovaný a každý PDO je přenášen v jednom rámci CAN s identifikátorem s vysokou prioritou. Z toho vyplývá,

že délka jednoho PDO může být maximálně 8 bajtů. Při požadavku přenosu většího množství aplikačních dat je nutné použít více samostatných PDO.

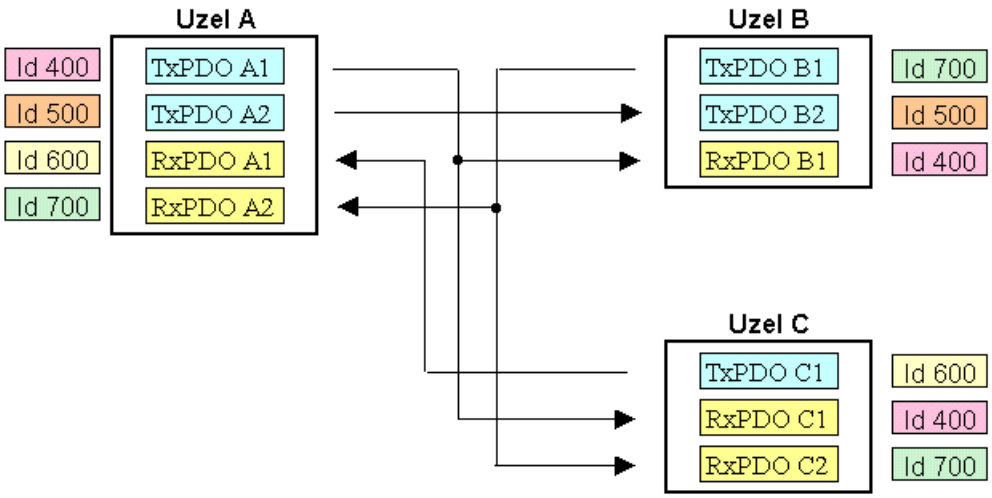
Každé zařízení umožňuje **přijímat až 256 procesních datových objektů** (ozn. RxPDO) **a vysílat až 256 procesních datových objektů** (ozn. TxPDO). Vysílané procesní datové objekty může přijímat jakákoliv okolní stanice. Komunikace PDO je zajištěna přidělením odpovídajících identifikátorů do příslušných komunikačních parametrů pro jednotlivé procesní datové objekty . Přijímací PDO, do kterých mají být předávána data z vysílacích PDO, musí mít shodnou hodnotu identifikátoru s identifikátorem vysílacího PDO.

Přípustné hodnoty identifikátorů pro jednotlivé kanály PDO jsou celá čísla v rozsahu, který je uveden v tab. 3.

PDO Identifikátor	Číselná soustava		
	desítková	šestnáctková	dvojková
min	384	180	00110000000
max	1023	3FF	01111111111

tab. 3: Rozsah povolených hodnot identifikátorů pro PDO

Je-li **hodnota identifikátoru** procesního datového objektu **nulová**, pak **není** tento procesní datový objekt **použit** pro přenos dat.



obr. 2: Příklad komunikace procesních datových objektů

4.4.2 Mapování PDO

Přiřazení aplikačních dat do rámce procesního datového objektu je zajištěno tzv. **PDO mapováním**. Každý PDO má vlastní strukturu mapovacích parametrů. Jednotlivé mapovací parametry jsou záznamy v objektovém slovníku umístěné na indexu, který odpovídá příslušnému PDO. Konkrétní umístění mapovacích parametrů každého procesního datového objektu je znázorněno v tab. 4.

	Index	Popis
Přijímací PDO	0100	map. parametry 1. RxPDO
	0101	map. parametry 2. RxPDO
	0102	map. parametry 3. RxPDO
	.	
	.	
	01FF	map. parametry 256. RxPDO
Vysílací PDO	0200	map. parametry 1. TxPDO
	0201	map. parametry 2. TxPDO
	0201	map. parametry 3. TxPDO
	.	
	.	
	02FF	map. parametry 256. TxPDO

tab. 4: Umístění mapovacích parametrů v objektovém slovníku

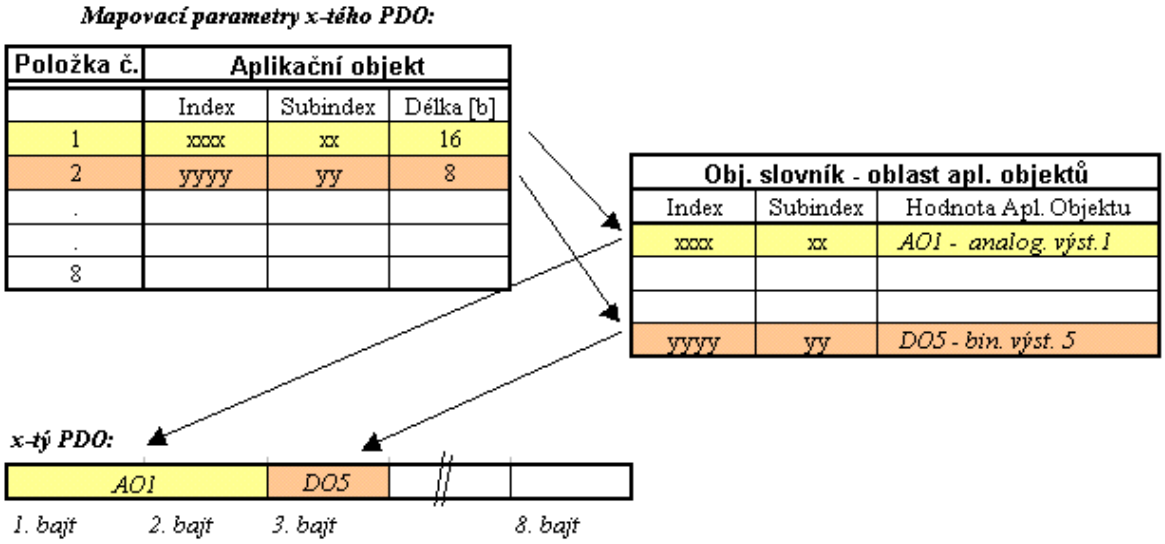
Jednotlivé mapovací parametry mohou mít maximálně osm položek. Každá z položek je uložena na příslušném indexu podle mapovaného PDO a na subindexu nula až sedm podle čísla položky. Tyto mapovací parametry informují o tom, na jakém indexu a subindexu v objektovém slovníku se nachází aplikační objekt, který má být namapován do příslušného PDO a dále jakou má velikost v bitech.

Název	Velikost	Obsah
Index	16b	Index mapovaného apl. objektu
Subindex	8b	Subindex mapovaného apl. objektu
Velikost	8b	Velikost aplikačního objektu v bitech

tab. 5: Formát mapovacího parametru

Na obr. 3 je názorný příklad mapování aplikačních objektů do PDO. Struktura mapovacího parametru x-tého PDO, která je umístěna v objektovém slovníku na

příslušném indexu podle pořadového čísla PDO (viz tab. 5) obsahuje konkrétní index a subindex v objektovém slovníku z oblasti aplikačních objektů. Na tomto indexu a subindexu je umístěn aplikační objekt, jehož obsah má být přenášen v x-tém PDO. Kromě hodnoty indexu a subindexu aplikačního objektu je ještě ve struktuře mapovacího parametru informace o délce aplikačního objektu v bitech.



obr. 3: Příklad mapování procesních datových objektů

Jednotlivé aplikační objekty jsou do rámce PDO vkládány za sebou podle umístění v mapovacích parametrech od subindexu nula. V případě, že jsou namapovány aplikační objekty o celkové délce překračující osm bajtů jsou všechna data nad tuto hranici ignorována, protože každý PDO může přenášet maximálně data o délce osmi bajtů. Proto je při mapování aplikačních objektů do PDO nutné dbát na to, aby nebyla překročena maximální povolená délka aplikačních objektů.

Mapované **aplikační objekty**, které jsou **větší než jeden bajt**, jsou v datovém poli PDO za sebou umísťovány v pořadí **od nejvíce významného po nejméně významný bajt**.

Pokud má jeden procesní datový objekt přijímat více okolních uzlů, není bezpodmínečně nutné, aby uzly svým výstupům přiřadily všechny aplikační objekty obsažené v přijatém PDO. Pokud určitý aplikační objekt přenášený v PDO není určen pro daný uzel, pak položky v mapovacích parametrech, které odpovídají pozici tohoto aplikačního objektu musí být nulové. Každá z osmi položek mapovacích parametrů,

která má index a subindex vyplněn nulami způsobí vynechání **jednoho bajtu** v datové oblasti PDO.

Příklad: Uzel přijme jeden procesní datový objekt, který ve svém datovém poli obsahuje: jeden aplikační objekt o délce dvou bajtů a za ním jeden aplikační objekt o délce jednoho bajtu. Výstupům tohoto uzlu se má přiřadit pouze druhý aplikační objekt. Proto první dva záznamy v mapovacích parametrech tohoto přijímacího PDO musí být vyplněny nulami. Až třetí záznam mapovacího parametru (na subindexu 2) bude obsahovat index a subindex konkrétního aplikačního objektu, který má být namapován. Do položky „délka“ je zapsána číslice osm (namapovaný aplikační objekt má délku jednoho bajtu). Další záznamy mapovacích parametrů jsou nulové.

4.4.3 Komunikační parametry

Způsoby přenosu a identifikátory procesních datových objektů jsou definovány pomocí tzv. **komunikačních parametrů**. Každý PDO má kromě záznamů s mapovacími parametry i svůj vlastní záznam s komunikačními parametry. Struktury s komunikačními parametry jsou uloženy v objektovém slovníku na příslušných indexech podle tab. 6.

	Index	Popis
Přijímací PDO	0300	kom. parametry 1. RxPDO
	0301	kom. parametry 2. RxPDO
	0302	kom. parametry 3. RxPDO
	.	
	.	
	03FF	kom. parametry 256. RxPDO
Vysílací PDO	0400	kom. parametry 1. TxPDO
	0401	kom. parametry 2. TxPDO
	0401	kom. parametry 3. TxPDO
	.	
	.	
	04FF	kom. parametry 256. TxPDO

tab. 6: Umístění komunikačních parametrů v objektovém slovníku

V každém z komunikačních parametrů příslušného PDO je odpovídající identifikátor, kód charakterizující způsob přenosu a dodatečný kód.

Název	Velikost	Obsah
Identifikátor	16b	Identifikátor PDO
Typ přenosu	8b	Zakódovaný typ přenosu PDO
Kód přenosu	8b	Dodatečný kód pro způsob přenosu PDO

tab. 7: Formát komunikačního objektu

První záznam komunikačního parametru obsahuje 16-bitovou položku, ve které je uložen **identifikátor** příslušného PDO. Tento identifikátor může nabývat hodnot od 384 (180 hex) do 1023 (3FF hex). Je-li hodnota identifikátoru nula, potom PDO, ke kterému přísluší daný komunikační parametr, není použit pro přenos procesních dat. V položce „**typ přenosu**“ je 8-bitový kód, který charakterizuje způsob přenosu konkrétního procesního datového objektu. Přípustné hodnoty pro tuto položku jsou:

- 01 - synchronní přenos.
- 02 - přenos v závislosti na přijetí žádosti jiného uzlu.
- 03 - přenos v závislosti na výskytu vnitřní události.

Posledním záznamem v komunikačních parametrech je **dodatečný kód** přenosu. Ten je v případě **vysílacího** PDO použit při přenosu vyvolaného vnitřní událostí a obsahuje dolní polovinu indexu dané události (viz kap. 4.3.2). V případě jiného způsobu přenosu na hodnotě tohoto kódu nezáleží. V komunikačním parametru **přijímacího** PDO má tento dodatečný kód přenosu význam, pouze pokud má být vyslán rámec žádosti o data jako reakce na událost. Potom tento kód musí obsahovat dolní polovinu indexu události, při jejímž výskytu bude vyslán rámec žádosti o data. Obdobně jako v případě vysílacího PDO při jiném způsobu přenosu na tomto kódu nezáleží.

4.4.4 Způsoby přenosů procesních datových objektů

V navrhovaném aplikačním protokolu je umožněno přenášet procesní data třemi způsoby:

- Synchronně.
- Na žádost jiného uzlu.
- V závislosti na výskytu vnitřní události v zařízení.

Synchronní přenos znamená, že daný PDO bude vyslán bezprostředně po přijetí speciálního synchronizačního objektu (SYNC). Tento synchronizační objekt je popsán v kap. 4.4.7.

Přenos na žádost jiného uzlu je uskutečněn po přijetí rámce žádosti o data (Remote Frame), jehož formát je definován ve specifikaci CAN 2.0A. Rámec žádosti může vyslat libovolná jednotka v síti. Přenos dat na žádost jiného uzlu se děje následovně:

Jednotka, která má přijímat procesní data tímto způsobem musí mít nastaven v komunikačních parametrech typ přenosu **přijímacího** PDO „na žádost“ (hodnota kódu 02) a do dodatečného kódu přenosu je nutné nastavit identifikátor události. Výskyt této události iniciuje vyslání rámce žádosti o data. Na tento rámec žádosti potom reagují všechny okolní jednotky, které mají identifikátor **vysílacího** PDO shodný s identifikátorem přijímacího PDO, který vysílá rámec žádosti. A typ přenosu mají nastaven „na žádost“ (hodnota kódu 02).

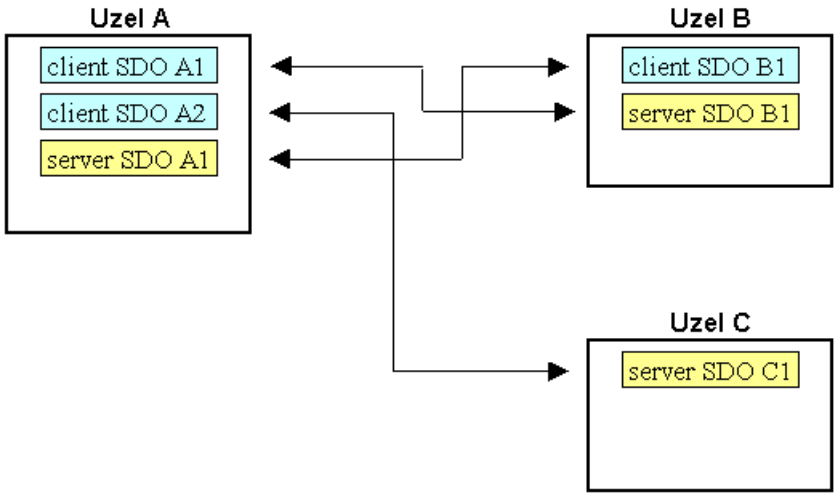
Vyslání PDO jako reakce **na vnitřní událost** se provede v případě, že se v daném zařízení vyskytne určitá událost, která je popsána identifikátorem události. Vysílací PDO musí mít v komunikačních parametrech nastaven typ přenosu „událost“ a v dodatečném kódu musí obsahovat nižší bajt konkrétního indexu události, která když nastane, pak je PDO vyslán. Tento procesní datový objekt mohou přijmout všechny okolní stanice, které mají u svých přijímacích PDO shodný identifikátor s vysílacím PDO a typ přenosu nastaven na „událost“, popřípadě i na „synchronní přenos“. Pokud by ovšem typ přenosu přijímacího PDO byl nastaven „na žádost“, potom by procesní datový objekt byl vysílán nejen při výskytu události ve vysílacím uzlu, ale i na žádost přijímacího uzlu.

4.4.5 Přenos konfiguračních dat

Konfigurační data jsou označována též jako tzv. **servisní datové objekty** (**SDO**). Tyto objekty mají přístup k záznamům v objektovém slovníku. Je umožněno z něho číst nebo do něho zapisovat hodnoty. Přenos těchto dat je na rozdíl od PDO potvrzovaný podle modelu client-server. Tyto objekty jsou určeny pro vzdálenou konfiguraci zařízení, k nastavování a čtení jeho mapovacích a konfiguračních parametrů procesních datových objektů. Dále je umožněno prostřednictvím SDO přečíst identifikátor modulu.

Klient je uzel, který komunikaci začíná v případě, že je potřeba přečíst nebo modifikovat data v objektovém slovníku jiného uzlu označovaném jako SDO server. Aby bylo umožněno zařízení konfigurovat, musí být toto zařízení **povinně vybaveno** právě jedním **SDO serverem**.

Přenos SDO probíhá v tzv. SDO kanálech tato komunikace je názorně zobrazena na obr. 4.



obr. 4: Příklad komunikace servisních datových objektů

Každý SDO kanál tvoří dva rámce s odlišným identifikátorem. Jeden ve směru client-server, který obsahuje příkaz a data. A druhý v opačném směru server-client, který nese potvrzení o úspěšnosti nebo neúspěšnosti prováděného příkazu.

Identifikátor ve směru client-server má binární tvar:

100 0XXX XXXX.

Identifikátor ve směru server-client má binární tvar:

110 0XXX XXXX.

V těchto binárních tvarech posledních sedm bitů představuje identifikátor modulu s SDO serverem .

Pomocí SDO rámců je možné číst záznamy z indexů 0100 až 02FF hex (mapovací parametry jednotlivých PDO) a z indexů 0300 až 04FF hex (komunikační parametry jednotlivých PDO). Zapisovat do objektového slovníku je povoleno na indexy 0100 až 02FF hex (mapovací parametry) a na indexy 0300 až 04FF hex (komunikační parametry).

Komunikace probíhá tak, že stanice s SDO clientem vyšle příkaz, který stanice s SDO serverem přijme, zpracuje a odešle potvrzovací zprávu, ve které je potvrzovací kód o úspěšnosti či neúspěšnost vykonávaného příkazu. V případě, že šlo o příkaz čtení, jsou spolu s potvrzovacím kódem přenášena ještě data z objektového slovníku.

Každý SDO příkaz je přenášen v jednom rámci CAN v následujícím formátu:

1. B	2.B	3.B	4.B	5.B	6.B	7.B	8.B
kód	Index H	Index L	Subindex	A	B	C	D

obr. 5: Formát SDO příkazu

1.B: kód příkazu: 10 hex = zápis do objektového slovníku;
20 hex = čtení z objektového slovníku.

2. B Index H: horních 8 bitů indexu záznamu ke čtení/zápisu.

3. B Index L: dolních 8 bitů indexu záznamu ke čtení/zápisu.

4. B Subindex: subindex záznamu ke čtení/zápisu.

5. až 8. B: formát závislý podle operace čtení/zápis a podle umístění čteného nebo zapisovaného objektu;

- čtení z objektového slovníku: na hodnotách A až D nezáleží.
- zápis do objektového slovníku: hodnoty A až D osahují zapisovaná data do objektového slovníku podle příslušného umístění (viz tab.8).

Index		A.	B.	C.	D.
0100-02FF	mapovací parametry	Index H	Index L	Subindex	Délka
0300-04FF	komunikační parametry	Id. H	Id L	Typ přenosu	Kód

tab. 8: Formát parametrů SDO příkazu

Formát SDO odpovědí:

1. B	2.B	3.B	4.B	5.B	6.B	7.B	8.B
kód	A	B	C	D	E.	F.	G.

obr. 6: Formát SDO odpovědi

1.B: kód odpovědi: 11 hex = zápis do objektového slovníku proveden;
12 hex = zápis do objekt. slovníku neproveden;
21 hex = čtení z objektového slovníku v pořádku;
22 hex = chyba při čtení z objektového slovníku;
FF hex = chybný formát SDO příkazu.

2. až 8. B: v případě, že čtení proběhlo v pořádku mají hodnoty A až G formát, který je uveden v tab. 9. V jiných případech na těchto hodnotách nezáleží a mohou mít libovolné hodnoty.

Index [hex]		A.	B.	C.	D.	E.	F.	G.
0100-02FF	mapovací par	Index H	Index L	Subindex	Index H	Index L	Subindex	Délka
0300-04FF	komunikační par.	Index H	Index L	Subindex	Id H	Id L	Typ přenosu	Kód

tab. 9: Formát parametrů SDO odpovědi

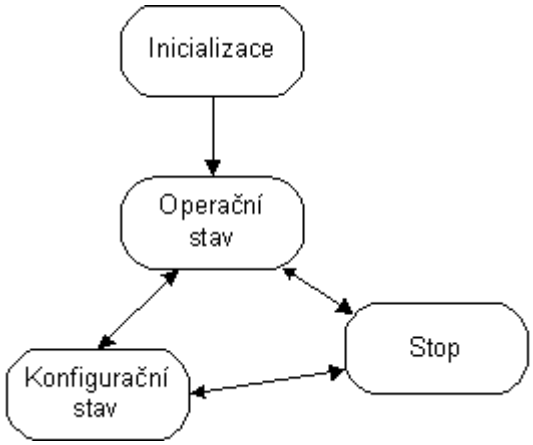
V případě odpovědí na příkaz ke čtení mapovacích nebo komunikačních parametrů je v polích označených A až C umístění čtených položek v objektovém slovníku, tak jak toto umístění bylo uvedeno v SDO příkazu. V polích označených E až G jsou hodnoty mapovacích nebo komunikačních parametrů, které byly přečteny z objektového slovníku.

Pokud byly čteny mapovací parametry, potom v polích D a E je uložena horní a dolní polovina indexu mapovaného aplikačního objektu, pole F obsahuje subindex a pole G nese informaci o délce mapovaného objektu v bitech.

V případě, že byly čteny hodnoty z indexů, na kterých se nalézají komunikační parametry procesních datových objektů, potom pole D obsahuje horní 3 bity a pole E obsahuje dolních 8 bitů 11-bitového identifikátoru daného PDO. V poli F je obsažen kód typu přenosu a pole G obsahuje dodatečný kód komunikačních parametrů.

4.4.6 Objekty spojené se správou sítě

Jsou definovány celkem čtyři stavy, ve kterých se zařízení může nacházet.



obr. 7 Stavy zařízení

V inicializačním stavu je zařízení bezprostředně po resetu. Po skončení inicializačních procedur se zařízení automaticky přepne do operačního stavu. V operačním stavu je umožněno vysílat a přijímat procesní datové objekty. Konfigurační stav umožňuje pouze komunikaci prostřednictvím servisních datových objektů. Pokud je zařízení v režimu „Stop“, což znamená, že je zastaveno, není umožněn přenos ani servisních ani procesních datových objektů.

Mezi posledními třemi z těchto stavů lze vzájemně přecházet pomocí tzv. NMT příkazů. To jsou speciální objekty, které jsou přenášeny jedním CAN rámcem s identifikátorem 100 (80 hex) a délkou datového pole dva bajty. Tento rámec může vyslat jakákoliv jednotka v síti a tím řídit stavy okolních jednotek. První bajt vždy obsahuje identifikátor modulu, pro který je příkaz určen a ve druhém bajtu je parametr příkazu podle tab. 10.

kód	příkaz
0x01	konfigurace
0x02	start
0x03	stop

tab. 10: Kódy NMT příkazů

4.4.7 Synchronizační objekt

Synchronizační objekt (ozn. SYNC) je přenášen v jednom datovém rámci CAN s identifikátorem 128 (80 hex) a s nulovou délkou datového pole. Generuje ho jednotka „Master“ v pravidelných časových okamžicích s periodou 100ms. Tento objekt slouží ke globální synchronizaci zařízení v síti. To znamená, že bezprostředně po přijetí synchronizačního objektu jednotka vyšle všechny procesní datové objekty, které mají v komunikačních parametrech nastaven synchronní typ přenosu.

5 Implementace navrženého protokolu

Navrhovaný aplikační protokol byl implementován do komunikačního uzlu s protokolem CAN na bázi mikrořadiče Atmel AT89C52, který byl realizován v roce 2003 v rámci diplomové práce (viz [1]) a do modulu C167CR/CAN Evaluation And Training Kit od firmy Hitex Ltd. s mikrořadičem Infineon C167CR.

Programy pro oba moduly byly vytvořeny v jazyku C ve vývojovém prostředí µVision2 od společnosti Keil Elektronik GmbH.

Dále byla pro CAN uzel s mikrořadičem AT89C52 vytvořena konfigurační aplikace pro PC. Použitím této aplikace je možné pomocí osobního počítače modifikovat parametry aplikačního protokolu CAN uzlu. Komunikace mezi osobním počítačem a uzlem s mikrořadičem AT89C52 probíhá po sériovém rozhraní RS-232C, kterým je standardně vybaven každý osobní počítač kompatibilní s IBM PC.

5.1 Modul na bázi AT89C52

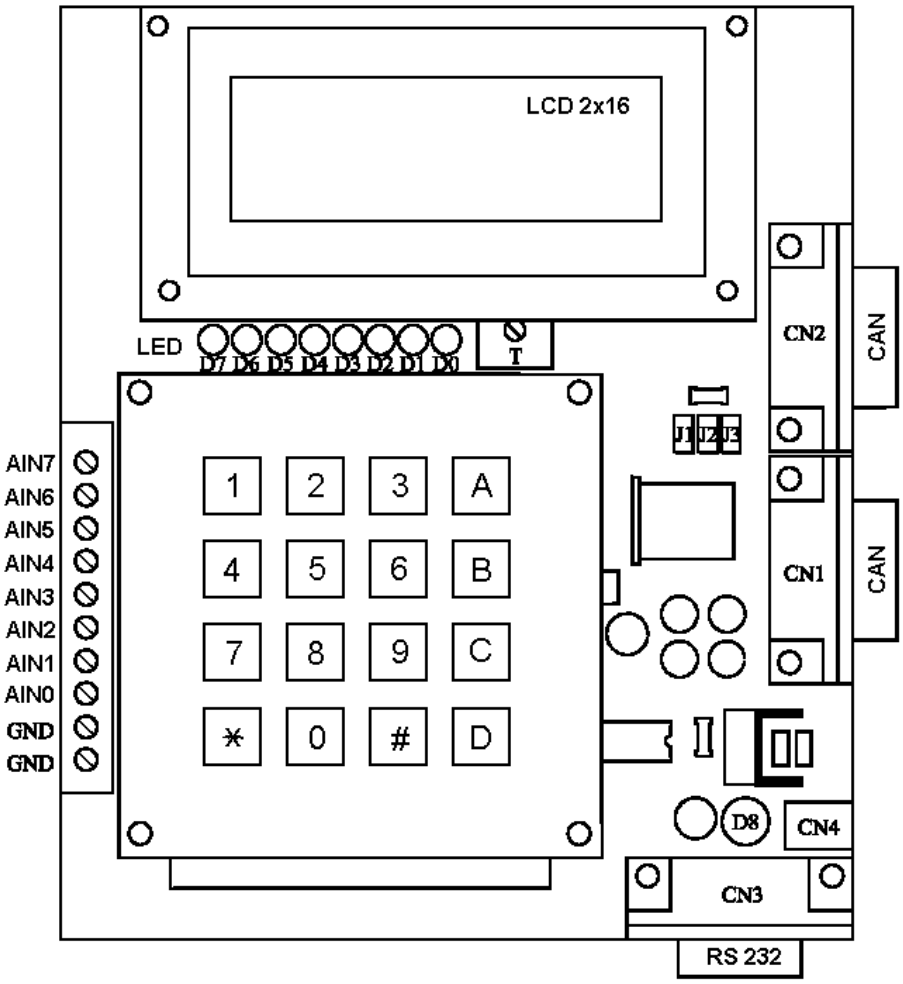
Tento modul, pro který je navržený aplikační protokol určen, je připojen na sběrnici CAN a plní funkci podřízené jednotky (Slave).

CAN uzel je schopen vysílat dva procesní datové objekty (TxPDO) a přijímat dva procesní datové objekty (RxPDO). Podle definice aplikačního protokolu je rovněž vybaven SDO Serverem, pomocí kterého je umožněna konfigurace po sběrnici CAN. **Identifikátor modulu** je programově nastaven na **10** (0A hex).

5.1.1 Stručný popis modulu

Jádro tvoří 8-bitový mikrořadič Atmel AT89C52 s datovou pamětí o velikosti 256 B a programovou pamětí Flash EEPROM o velikosti 8 kB. K mikrořadiči je přes sběrnici I^2C připojena napěťově nezávislá paměť EEPROM Atmel AT24C04 o velikosti 512 B. Přístup na sběrnici CAN je řešen pomocí řadiče CAN Intel 82527 a budiče sběrnice Philips PCA82C250. Uzel je dále vybaven osmikanálovým 12-bitovým AD převodníkem Analog Devices AD7890-10, maticovou klávesnicí se šestnácti tlačítky,

osmi LED, dvouřádkovým maticovým LCD displejem a sériovým rozhraním RS-232C. Podrobný popis modulu je uveden v [1].



obr. 8: Uspořádání modulu

5.1.2 Uživatelské rozhraní

Pro zajištění rozhraní mezi uživatelskými vstupy, výstupy a sběrnicí CAN jsou podle navrženého protokolu určeny **aplikační objekty** (viz kap. 4.3.3). CAN uzel umožňuje vysílat a přijímat devět aplikačních objektů, které je možné mapovat do dvou přijímacích a dvou vysílacích procesních datových objektů. Umístění jednotlivých vstupů a výstupů (aplikačních objektů) v objektovém slovníku a jejich velikosti jsou uvedeny v tab. 11.

	Název	Index [hex]	Subindex [hex]	Velikost [b]
Vstupy	Analog. vstup AIN0	0700	00	16
	Analog. vstup AIN1	0700	01	16
	Analog. vstup AIN2	0700	02	16
	Analog. vstup AIN3	0700	03	16
	Analog. vstup AIN4	0700	04	16
	Analog. vstup AIN5	0700	05	16
	Analog. vstup AIN6	0700	06	16
	Analog. vstup AIN7	0700	07	16
	Kód klávesy	0710	00	8
Výstupy	LED	0800	00	8
	LCD 1. řádek 1.pozice	0810	00	8
	LCD 1. řádek 2.pozice	0810	01	8
	LCD 1. řádek 3.pozice	0810	02	8
	LCD 1. řádek 4.pozice	0810	03	8
	LCD 2. řádek 1.pozice	0811	00	8
	LCD 2. řádek 2.pozice	0811	01	8
	LCD 2. řádek 3.pozice	0811	02	8
	LCD 2. řádek 4.pozice	0811	03	8

tab. 11: Umístění aplikačních objektů v objektovém slovníku

Vstupy modulu:

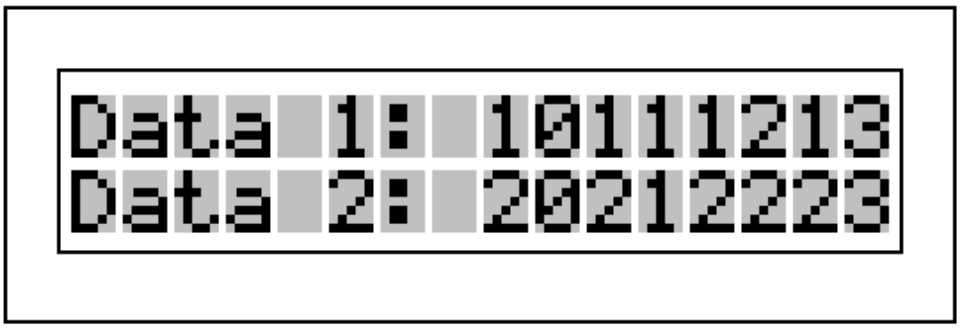
- *Analogové vstupy* (svorky AIN0 až AIN7): jsou vstupy analogově-digitálního převodníku, které mají v oblasti aplikačních objektů přiřazen jednotný index 0700 hex a subindex odpovídající číslu vstupu. Obsah jednotlivých aplikačních objektů odpovídá hodnotě napětí příslušného kanálu AD převodníku. Každý kanál je snímán jednou za osmset milisekund (vzorkovací frekvence 1,25Hz). Délka aplikačního objektu je 16 bitů, z toho jsou čtyři nejvíce významné bity vyplněny nulami a zbylých dvanáct bitů odpovídá navzorkované hodnotě napětí.
- *Kód klávesy:* v tomto aplikačním objektu se uchovává 8-bitový binární kód naposledy stisknuté klávesy.

Výstupy modulu:

- *LED:* tento aplikační objekt je umístěn v objektovém slovníku na indexu 0800 hex a subindexu 0, jeho délka je osm bitů. Každý bit tohoto aplikačního objektu ovládá jednu z osmi LED diod. Nejméně významný bit odpovídá LED diodě na obr. 8 označené D0 a nejvíce významný bit odpovídá LED

diodě označené D7. LED diody budou svítit v případě, že odpovídající bity v tomto aplikačním objektu budou v log. 1.

- *LCD:* vzhled displeje v operačním režimu je vyobrazen na obr. 9. Na každém řádku je text „Data“, číslo řádku a dvojtečka. Na dalších pozicích za tímto textem je pole, kde je možné zobrazit čtyři hexadecimální čísla (00 až FF). Každá pozice zobrazovaného hexadecimálního čísla je představována jedním aplikačním objektem o délce osmi bitů v objektovém slovníku. Aplikační objekty čísel zobrazovaných na prvním řádku mají společný index 0810 hex a subindex od 00 do 03 podle pořadí zobrazované číslice. Čísla na druhém řádku odpovídají aplikačním objektům na indexu 0811 hex a subindexech 00 až 03 v pořadí jako u čísel na prvním řádku.



obr. 9: Vzhled displeje v operačním režimu

5.1.3 Mikrořadič Atmel AT89C52

Inicializace a obsluha mikrořadiče AT89C52 je naprogramována v modulu *can_a.c*.

Atmel AT89C52 je nízkopříkonový vysoce výkonný 8-bitový mikrořadič vyrobený technologií CMOS. Je vybaven vnitřní datovou pamětí o velikosti 256 B a programovou pamětí Flash s kapacitou 8 kB. Tento mikrořadič je vývody a instrukčním souborem kompatibilní s průmyslovými standardy 80C51 a 80C52. Obsahuje tři 16-bitové čítače/časovače, programovatelný sériový kanál a 32 vstupně/výstupních linek uspořádaných ve čtyřech portech (P0 až P3). Frekvence hodinového signálu může být 0 až 24 MHz.

Datová paměť je určena pro ukládání proměnných a pro zásobník. Dolní polovina datové paměti (00H – 07H) je přístupná přímým i nepřímým adresováním. Horní polovina datové paměti (08H - FFH) je přístupná pouze nepřímým adresováním, protože se tato oblast překrývá s oblastí speciálních funkčních registrů (SFR). V jazyce Cx51 se pro nepřímý přístup do datové paměti používá klíčové slovo *idata*. Například má-li být proměnná *prom* typu *unsigned char* uložena v horní polovině datové paměti, potom musí být deklarována následovně:

```
unsigned char idata prom;
```

Oblast speciálních funkčních registrů je horní polovina datové paměti. Ta je přístupná přímým adresováním (část této oblasti i bitovým adresováním) a obsahuje důležité informace o stavech procesoru, jeho periferních obvodů a informace, které ovlivňují vlastnosti procesoru.

Dva 16-bitové čítače/časovače T0 a T1 mohou pracovat ve čtyřech módech, které se nastavují v registru TMOD. Horní polovina tohoto registru slouží pro konfiguraci č/č T1 a dolní polovina pro č/č T0.

- **mód 0**: 8-bitový č/č tvořen registrem THn s předřazeným 5-bitovým předděličem tvořeným registrem TLn.
- **mód 1**: 16-bitový č/č tvořen registry THn a TLn.
- **mód 2**: 8-bitový čítač s přednastavením. Registr TLn je obsah čítače a při přetečení se nastaví příznak TFn a TLn se naplní předvolbou v THn.
- **mód 3**: č/č T0 je rozdělen na dva samostatné 8-bitové č/č TH0 a TL0. Je-li T0 v módu 3, potom lze č/č T1 využít jen pro generování rychlosti sériového kanálu nebo pro č/č bez možnosti přerušení.

Čítač/časovač T2 má vlastní konfigurační registr T2CON a může pracovat ve třech módech:

- **mód 0**: 16-bitový č/č s automatickým přednastavením. V případě přetečení registrů TH2 a TL2 se nastaví příznak TF2 a tyto dva registry se nastaví předvolbami v RCAP2H a RCAP2L.
- **mód 1**: Záchytný režim. Je-li EXEN2 = 0, pak T2 pracuje jako 16-bitový č/č. Je-li EXEN2 = 1, pak T2 pracuje stejně jako v předchozím případě,

ale při sestupné hraně na vstupu T2EX dojde k uložení aktuálních hodnot TH2 a TL2 do RCAP2H a RCAP2L.

- **mód 2**: Generátor rychlosti sériového kanálu.

Sériový kanál se konfiguruje hodnotami v registru SCON. Pro vysílání a přijímání dat slouží vyrovnávací registr SBUF. Je-li povoleno přerušení od sériového kanálu, je při přijetí dat nastaven příznak RI a při odvysílání nastaven příznak TI. Tyto příznaky je nutné nulovat programově.

Jsou umožněny čtyři módy sériového přenosu:

- **mód 0**: 8-bitový synchronní přenos s pevnou přenosovou rychlostí $f_{osc}/12$.
- **mód 1**: 8-bitový asynchronní přenos s prog. přenosovou rychlostí.
- **mód 2**: 9-bitový asynchronní přenos s volitelnou rychlostí přenosu.
- **mód 3**: 9-bitový asynchronní přenos s prog. přenosovou rychlostí.

AT89C52 má šest zdrojů přerušení, z toho dva externí, tři od čítačů/časovačů a jeden zdroj od sériového kanálu. Každý z těchto zdrojů může být nezávisle na ostatních povolen či zakázán bitem povolujícím přerušením v registru IE. Tento registr navíc obsahuje bit EA, který globálně povoluje nebo zakazuje všechny zdroje přerušení. Prioritní úroveň lze změnit pomocí registru IP. Další potřebné informace lze nalézt např. v [4] nebo [11].

5.1.4 Řadič CAN Intel 82527

Řadič sériové komunikace Intel 82527 poskytuje přístup na rozhraní CAN. Podporuje specifikaci CAN 2.0 ve standardním i rozšířeném formátu. Má programovatelnou masku pro standardní i rozšířený identifikátor zpráv. Obsahuje celkem 14 objektů zpráv (Message 1 až 14) pro vysílání i příjem a jeden objekt zpráv (Message 15) s vlastní programovatelnou maskou, který je určen pouze pro příjem rámců. Rychlost přenosu je programovatelná. Dále je vybaven přerušovacím systémem, který umožňuje generovat přerušení při příjmu či vyslání zprávy nebo při detekci chyby.

Přerušení vyvolané výskytem chyby může nastat dvěma způsoby. V prvním případě pokud vnitřní čítač chyb CAN řadič dosáhne hodnoty 96, je nastaven

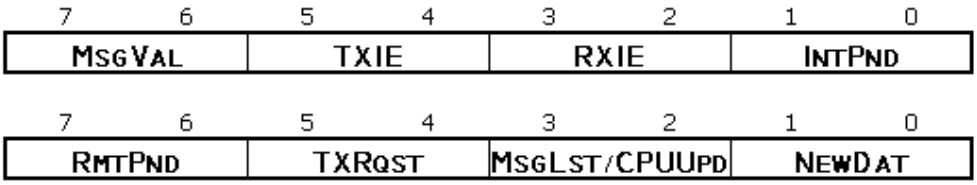
tzv. Warning Status. Ve druhém případě, pokud vnitřní čítač chyb dosáhne hodnoty 256, je nastaven tzv. Bus Off Status a řadič CAN je odpojen od sběrnice. Oba chybové stavy mohou vyvolat přerušení, pokud jsou v řídicím registru (Control Register) nastaveny bity IE a EIE.

Řadič 82527 dále obsahuje dva 8-bitové obousměrné porty pro připojení periferií. Všechny konfigurační registry, jednotlivé objekty zpráv a registry obou portů jsou umístěny ve vnitřní paměti RAM. Konkrétní rozmístění jednotlivých registrů v paměti RAM je uvedeno v [5].

Struktura objektů zpráv je vidět na obr. 10.

obr. 10: Struktura objektů zpráv

Každý objekt zpráv obsahuje dva řídicí registry Control 0 a Control 1 na adresách Base + 0 a Base + 1. Položky těchto registrů přímo řídí přenos CAN rámců a umožňují konfigurovat přerušení zpráv.



obr. 11: Řídicí registry Control 0 a Control 1

MsgVal: Validace objektů. 1: objekt je platný. 0: objekt je neplatný a nelze vysílat ani přijímat data.

TXIE: Povolení přerušení při vyslání zprávy. 1: při bezchybném odvysílání řadič generuje přerušení.

RXIE: Povolení přerušení při přijetí zprávy. 1: při přijetí zprávy řadič generuje přerušení.

IntPnd: Nevyřízené přerušení. 1: tento objekt vyvolal přerušení. 0: od posledního vynulování nebylo generováno přerušení.

RmtPnd: Nevyřízená odpověď na rámec žádosti. 1: Byl přijat rámec žádosti, ale ještě nebyla odeslána odpověď.

TxRqst: Vyslání rámce. 1: Požadavek na vyslání rámce.

MsgLst: Ztráta zprávy (pouze pro přijímací objekty). 1: Byl přijat nový rámec, když byl nastaven příznak NewDat.

CPUUpd: Aktualizace dat (pouze pro vysílací objekty). 1: rámec nemůže být vyslán.

NewDat: Nová data o datovém poli objektu.

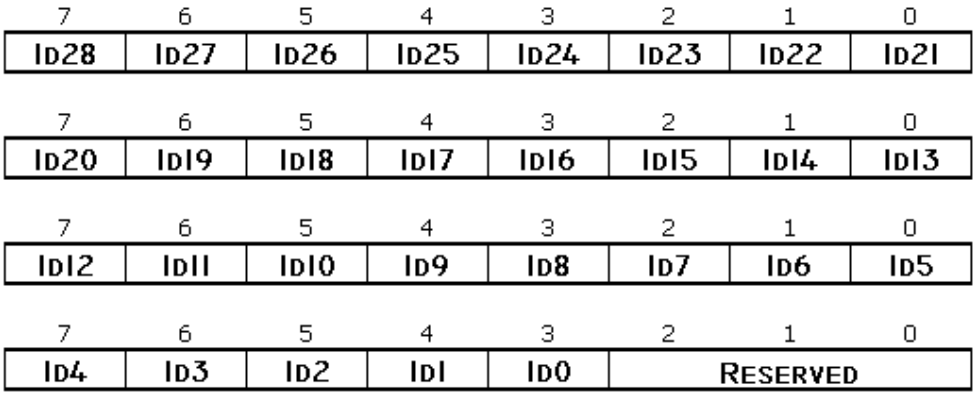
Příznaky těchto dvou registrů se nastavují jiným způsobem, než je obvyklé u jiných registrů. Každý příznak se skládá ze dvou bitů, jejichž kombinací se do příznaku zapisuje nebo čte. Možné kombinace jsou:

	MSB	LSB	Význam
Zápis	0	0	nepovoleno
	0	1	nulování
	1	0	nastavení
	1	1	beze změny
Čtení	0	1	vynulován
	1	0	nastaven

tab. 12: Kombinace pro operace s řídícími registry

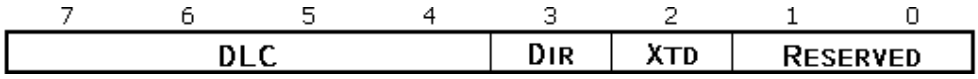
Identifikátor zpráv jednotlivých objektů je uložen v registrech Arbitration 0 až 3 na adresách Base + 2 až Base + 5. Rozmístění jednotlivých bitů je uvedeno na obr. 12. Rozšířenému 29-bitovému identifikátoru odpovídají bity Id 0 až Id 28. Pro standardní

11-bitový formát identifikátoru jsou určeny bity Id 18 až Id 28, na ostatních bitech nezáleží a jsou ignorovány.



obr. 12: Registry Arbitration 0 - 3

Pro konfiguraci jednotlivých objektů zpráv je určen registr Mess. Conf. (Message Configuration Register). Je umístěn na adrese Base + 6. Struktura tohoto registru je následující:



obr. 13: Registr Message Configuration

DLC: Délka datového pole. Platné hodnoty jsou 0 až 8.

DIR: Směr přenosu. Log. 1: vysílání. Log. 0: příjem.

XTD: Identifikátor. Log. 1: rozšířený 29-bitový identifikátor.
Log. 0: standardní 11-bitový identifikátor.

Datové pole tvořené registry Data 0 až Data 7 na adresách Base + 7 až Base + 14 slouží pro zápis dat vysílaných rámců nebo jako registry, do kterých se ukládají data přijatých rámců. Podrobný popis týkající se řadiče Intel 82527 lze nalézt v [5], [6] nebo v [7].

5.1.5 Propojení mezi mikrořadičem AT89C52 a řadičem 82527

Řadič 82527 je k mikrořadiči AT89C52 připojen v osmibitovém multiplexovaném režimu k portu P0 jako externí paměť dat o velikosti 256 bajtů. Dále jsou připojeny řídicí signály $\overline{WR}$ a $\overline{RD}$, signál $\overline{ALE}$, signál pro resetování radiče a signál $\overline{INT}$, který slouží jako zdroj externího přerušení INT0.

V programu se k jednotlivým registrům, které jsou umístěny v paměti RAM řadiče 82527, přistupuje pomocí předdefinovaného makropříkazu `PBYTE`, který je definován ve hlavičkovém souboru *absacc.h*. Tento hlavičkový soubor je součástí vývojového prostředí Keil µVision2 a je umístěn v adresáři *\C51\inc*. Makro `PBYTE` zajistí přímý přístup do externí paměti o velikosti jedné stránky (256B). Počáteční adresa této stránky je definována v inicializačním zdrojovém souboru *startup.a51*, který je rovněž součástí vývojového prostředí.

Použití makra `PBYTE` je názorně vidět na následujícím příkladu:

```
prom = PBYTE[0x01];
PBYTE[0x01] = prom;
```

V prvním případě se proměnná `prom` naplní hodnotou délky jednoho bajtu, která je uložena ve stránce vnější paměti na adrese 01 hex. Ve druhém případě se na adresu 01 hex stránky vnější paměti zapíše hodnota proměnné `prom`.

5.1.6 Softwarové vybavení

Program se skládá z následujících modulů a příslušných hlavičkových souborů:

startup.a51
main.c
can_a.c
eeprom.c
lcd.c
adc.c
keypad.c

Zdrojový souboru *startup.a51* obsahuje inicializaci vyvolanou po resetu mikrořadiče AT89C52. Je napsán v asembleru a je součástí vývojového prostředí µVision2. Modul *main.c* obsahuje volání inicializační funkce a uzavřenou programovou smyčku, ve které se podle globální proměnné `State` volá jedna ze tří funkcí v závislosti na požadovaném stavu uzlu.

V modulu *can_a.c* jsou inicializační funkce jednotlivých registrů mikrořadiče AT89C52 a řadiče CAN, funkce obsluhující jednotlivé stavy uzlu, obsluha komunikace CAN řadiče a další funkce aplikačního protokolu. Komunikace se sériovou pamětí po sběrnici I^2C je naprogramována v modulu *eeprom.c*. Programový modul *lcd.c* obsahuje funkce obsluhující LCD displej. Moduly *adc.c* a *keypad.c* obsahují obslužné rutiny AD převodníku a klávesnice. Veškeré zdrojové kódy jsou na přiloženém CD.

5.1.7 Globální proměnné

V horní oblasti datové paměti jsou uloženy struktury komunikačních a mapovacích parametrů obou přijímacích i obou vysílacích procesních datových objektů. Tyto parametry obsahují důležité údaje pro přenos procesních datových objektů (viz kap. 4.4.2 a 4.4.3). V aplikačním protokolu mají význam záznamů v objektovém slovníku a podle toho k nim lze i přistupovat. Definice struktur mapovacích a komunikačních parametrů jsou následující:

```
struct PDOMapRec
{
     unsigned char kod;
     unsigned char length;
};

struct PDOCommRec
{
     unsigned int  COB_ID;
     unsigned char Tr_Type;
     unsigned char Code;
 };
```

Tyto struktury jsou definovány v hlavičkovém souboru *can_a.h*. Každý procesní datový objekt (PDO) parametrizuje jedna struktura komunikačních parametrů a pole osmi struktur mapovacích parametrů. Z důvodu úspory místa v datové paměti a omezeného počtu aplikačních objektů jsou položky index a subindex v mapovacích parametrech konvertovány na tzv. vnitřní objekty, kde každá tato proměnná zabírá v datové paměti pouze jeden bajt místo původních tří bajtů (2B index + 1B subindex). Pro převody mezi

indexem, subindexem a tímto vnitřním kódem jsou určeny funkce `Code2Indx`, `Code2Sindx` a `Indx2Code`.

Dále jsou deklarovány globální proměnné pro aplikační objekty, které jsou rovněž definovány implementovaným aplikačním protokolem (viz kap. 4). Pole osmi proměnných typu `int`, do kterých jsou ukládány hodnoty napětí AD převodníku. Pole osmi proměnných typu `unsigned char`, které obsahují jednotlivé zobrazované hexadecimální čísla na LCD displeji. Jednu proměnnou typu `unsigned char`, podle které se rozsvěcejí LED diody a jednu proměnnou `unsigned char` pro ukládání kódu naposledy stisknuté klávesy.

V proměnné `State` je zakódován aktuální stav uzlu. V závislosti na této proměnné se v modulu *main.c* volají funkce, které odpovídají jednotlivým stavům uzlu.

Globální proměnná `disp` slouží pro uložení informace o tom, jaký textový řetězec se bude zobrazovat na LCD displeji.

Ostatní globální proměnné jsou určeny pro operace spojené s přenosem dat po sériové lince.

5.1.8 Stavy uzlu

Podle navrženého aplikačního protokolu se CAN uzel může nacházet v jednom ze čtyř stavů. Prvním stavem je inicializace uzlu, v tomto stavu se nachází bezprostředně po připojení napájení. Po skončení inicializace se uzel automaticky přepne do operačního stavu, ve kterém je umožněno vysílat a přijímat rámce procesních datových objektů podle nastavených komunikačních a mapovacích parametrů. Z tohoto stavu je možné přepnutí buď do konfiguračního stavu nebo zastavení uzlu. V konfiguračním stavu je možné modifikovat mapovací a komunikační parametry procesních datových objektů prostřednictvím příkazů servisních datových objektů nebo pomocí konfigurační aplikace pro osobní počítač, který je s uzlem propojen sériovou linkou. Pokud je uzel zastaven, není možné vysílat ani přijímat procesní ani servisní datové objekty. Přepínat mezi operačním a konfiguračním stavem nebo zastavit uzel lze dvěma způsoby. Prvním způsobem je přijetí CAN rámce od jiného uzlu (zpravidla Mastera) s NMT příkazem (viz kap. 4.4.6), v jehož datovém poli je obsažen identifikátor uzlu, pro který je příkaz určen a kód příkazu, kterým se uzel přepne do

požadovaného stavu. Druhou možností změny stavu je pomocí příkazu přes sériové rozhraní z řídicího počítače.

5.1.9 Inicializace

Po připojení napájecího napětí dojde k inicializaci uzlu. Během inicializace je na LCD displeji na prvním řádku zobrazen text: „Init“ a na druhém řádku je: „CAN Node v. 2.0“. Při inicializaci dochází k nastavení módů čítačů/časovačů, parametrů sériové linky, inicializaci LCD, AD převodníku a CAN řadiče. Pokud paměť EEPROM obsahuje platné mapovací a komunikační parametry PDO, jsou tyto parametry z paměti vyzvednuty a podle nich jsou parametrizovány jednotlivé objekty zpráv CAN řadiče. Konkrétní umístění příslušných položek v paměti je uvedeno v příloze. V případě, že paměť EEPROM neobsahuje platné hodnoty mapovacích a komunikačních parametrů, jsou procesní datové objekty parametrizovány podle výchozího nastavení, které je součástí řídicího programu CAN uzlu. Před skončením inicializačního procesu je povoleno externí přerušení na sestupnou hranu od vstupu INT0, ke kterému je připojen přerušovací výstup řadiče CAN 82527. Dále je povoleno přerušení od časovače T0 a přerušení od sériového kanálu.

V inicializační funkci se jako první nastavují parametry mikrořadiče AT89C52. Nejprve jsou nastaveny módy a předvolby všech tří čítačů/časovačů (T0 až T2), kterými je tento mikrořadič vybaven.

Čítač/časovač **T0** je provozován v režimu časovače, který je určen pro generování přerušení každých 50 ms. V programové obsluze přerušení od tohoto časovače jsou aktualizovány hodnoty aplikačních objektů vysílaných procesních datových objektů. Dochází k aktualizaci LED diod podle příslušné proměnné `LED` a k nastavení příznaku, pomocí něhož je v operačním režimu aktualizován LCD displej a hodnota jednoho kanálu AD převodníku. Čítač/časovač T0 je nastaven do módu 1, kde pracuje jako 16-bitový časovač s předvolbou v registrech TH0 a TL0. Pro hodnotu předvolby platí vztah:

$$[TH0, TL0] = 65536 - \frac{f_{osc}}{12 \cdot f_{T0}} = 65536 - \frac{11{,}0592 \cdot 10^6}{12 \cdot 5} = 19456 = 4C00hex.$$

Čítač/časovač **T1** slouží pro generování přenosové rychlosti sériového kanálu. Komunikační rychlost sériové linky je pevně stanovená na 9600 b/s. Pro tuto rychlost musí být T1 nastaven v módu 2 a pro předvolbu platí:

$$[TH0] = 256 - \frac{2^{SMOD} \cdot f_{osc}}{384 \cdot rychlost} = 256 - \frac{2^0 \cdot 11{,}0592 \cdot 10^6}{284 \cdot 9600} = 253 = FDhex.$$

Čítač/časovač **T2** je určen pro generování hodinového signálu pro AD převodník (vstup CLK IN). Pracuje v režimu 16-bitového časovače s automatickým přednastavením jako programovatelný generátor hodinového signálu s frekvencí 2,7648 MHz. Hodnoty registrů předvolby jsou nastaveny podle vztahu:

$$[RCAP2H, RCAP2L] = 65536 - \frac{f_{osc}}{4 \cdot f_{T2EX}} = 65536 - \frac{11{,}0592 \cdot 10^6}{4 \cdot 2{,}7648 \cdot 10^8} = 65535 = FFFEhex.$$

Dále jsou nastaveny parametry sériového kanálu, který slouží pro komunikaci s osobním počítačem. Pro nastavení sériové linky je určen registr SCON. Sériový kanál je nastaven do módu 1 (osmibitový přenos s programovatelnou přenosovou rychlostí). Přenos začíná nulovým start bitem, následuje osm datových bitů, přičemž první bit je bit s nejnižší váhou a končí jedničkovým stop bitem. Jako generátor rychlosti přenosu je určen časovač T1.

Potom jsou nastaveny řídicí signály AD převodníku. Je inicializován LCD displej a zobrazen inicializační text.

Dále jsou vynulovány struktury mapovacích a komunikačních parametrů. Pak je provedena kontrola, zda v paměti EEPROM jsou platné hodnoty mapovacích a komunikačních parametrů pro procesní datové objekty. To spočívá v tom, že je kontrolován záznam na adrese 01 v této paměti a pokud je tento záznam roven konstantě 01, jsou data v paměti považována za platná. Jsou-li hodnoty v EEPROM platné, potom jimi jsou naplněny mapovací a komunikační parametry procesních datových objektů. V opačném případě jsou do mapovacích a komunikačních parametrů zapsány hodnoty, které jsou součástí řídicího programu.

Po načtení konkrétních hodnot mapovacích a komunikačních parametrů dochází k inicializaci řadiče CAN. Ten je nejprve resetován, poté se ve smyčce čeká, dokud je

aktivní (v log. 1) bit RstSt (Hardware Reset Status) v registru CPUIR (CPU Interface Register). Potom jsou nastaveny bity Init a CCE v Control Registru. Nastavením bitu Init (Initialization) se povoluje softwarová inicializace řadiče, po dobu nastavení tohoto bitu jsou výstupy řadiče CAN v recesivní úrovni. Při nastaveném bitu CCE (Change Configuration Enable) je povoleno zapisovat do konfiguračních registrů řadiče. V registru CPUIR je nastaven bit DSC (Divide Systém Clock). Nastavením tohoto bitu je systémový hodinový signál (SCLK) roven polovině frekvence krystalu. Konfigurační registr portu P1, na který je připojena maticová klávesnice je nastaven tak, že horní čtyři bity portu jsou nastaveny jako výstup a dolní čtyři bity jako vstup. Port P2, na který jsou připojeny LED diody, je nastaven jako výstupní zapsáním hodnoty FFhex do konfiguračního registru P2CONF. Do registru BCR (Bus Configuration Register) je zapsána hodnota 48hex (nastaveny bity CoBy a DcT1). Nastavením bitu CoBy (Comparator Bypass) dojde k odstavení vstupního komparátoru řadiče a nastavením bitu DcT1 (Disconnect TX1 output) je zakázán ovladač výstupu TX1. Toto nastavení odpovídá způsobu připojení řadiče 82527 k mikrořadiči AT89C52. V registrech Bit Timing Register 0 a Bit Timing Register 1 jsou hodnoty, které odpovídají přenosové rychlosti na sběrnici CAN 100 kb/s a vzorkovacímu bodu 75 %. Postup výpočtu hodnot těchto registrů je uveden v [5]. Hodnota v registru Bit Timing Register 0 je 87 hex a hodnota v registru Bit Timing Register 1 je 25 hex. Registry globální masky objektů zpráv jsou naplněny hodnotami FFhex, což znamená, že si musí odpovídat všechny bity identifikátoru přijímaných zpráv. Po vynulování veškerých registrů všech objektů zpráv, jsou nastaveny konfigurační registry a identifikátory objektů zpráv, které jsou použity pro příjem synchronizačního objektu SYNC (viz kap. 4.4.7), pro příjem NMT příkazů (viz kap. 4.4.6) a dva objekty zpráv pro příjem a vysílání servisních datových objektů (viz kap. 4.4.5).

Objekt zpráv	Obsah	Identifikátor
Message 1	přijímací PDO1	384-1023 (180-3FF hex)
Message 2	přijímací PDO2	384-1023 (180-3FF hex)
Message 3	vysílací PDO1	384-1023 (180-3FF hex)
Message 4	vysílací PDO2	384-1023 (180-3FF hex)
Message 5	SDO Server - příjem příkazů	1034 (40A hex)
Message 6	SDO Server - vysílání odpovedí	1546 (60A hex)
Message 7	příjem NMT příkazů	100 (64 hex)
Message 8	příjem synchronizační objektu (SYNC)	128 (80 hex)

tab. 13: Obsazení jednotlivých objektů zpráv řadiče 82527

Pro příjem synchronizačního objektu SYNC byl zvolen objekt zpráv Message 8 (adr. v RAM: 80-8E hex). Tento objekt je nastaven jako přijímací s identifikátorem 128 a má povoleno vyvolat přerušení při přijetí synchronizačního rámce. Na přerušení od tohoto objektu se reaguje v operačním režimu uzlu při vysílání procesních datových objektů, které jsou konfigurovány na synchronní přenos.

Rámec s NMT příkazem, kterým lze přepínat stavy uzlu, je přijímán objektem zpráv Message 7 (adr. v RAM: 70-7E hex). Tento objekt je konfigurován podobně jako v předchozím případě, jen s tím rozdílem, že jeho identifikátor je 100. Rovněž může vyvolat přerušení při přijetí rámce s NMT příkazem. V obsluze přerušení od tohoto rámce se zpracuje NMT příkaz a podle něho se nastaví globální proměnná `State`, v závislosti na které se v modulu *main.c* volají funkce odpovídající jednotlivým stavům uzlu.

Objekty zpráv Message 5 a Message 6 jsou určeny pro příjem a vysílání rámců servisních datových objektů SDO Serveru. Pro příjem SDO příkazů je určen objekt zpráv Message 5, který je nastaven jako přijímací s identifikátorem 1034 (40A hex). A pro vysílání odpovědí na SDO příkazy slouží objekt Message 6, který je nastaven jako vysílací s délkou datového pole 8 B a s identifikátorem 1546 (60A hex). Postup sestavování identifikátorů pro servisní datové objekty je popsán v kap. 4.4.5.

Pro procesní datové objekty jsou objekty zpráv parametrizovány podle konkrétních komunikačních parametrů příslušného PDO. Identifikátor pro tyto objekty může být v rozmezí od 384 do 1023. Pokud je identifikátor určitého procesního datového objektu nulový, potom tento objekt není použit pro přenos a příznak validace objektu (MsgVal) v řídícím registru Control 0 není nastaven. Pro přijímací PDO jsou vyhrazeny objekty zpráv Message 1,Message2 a pro vysílací Message 3 a Message 4.

Před ukončením inicializace CAN řadiče jsou vynulovány bity řídicího registru CCE a Init. Tím končí softwarová inicializace řadiče 82527 a jsou nastaveny bity EIE i IE. Nastavením těchto dvou bitů je řadiči CAN povoleno generovat přerušení.

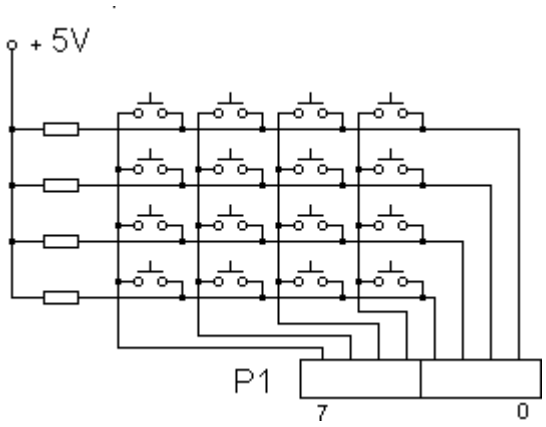
Dále je aktivováno přerušení mikrořadiče AT89C52 od časovače T0, od sériového kanálu a přerušení na sestupnou hranu od vnějšího zdroje přerušení, na který je připojen řadič Intel 82527.

Před koncem inicializační procedury se do globální proměnné `State` zapíše kód, který specifikuje operační stav uzlu, ve kterém se uzel bude nacházet po skončení inicializace.

5.1.10 Operační stav

V tomto stavu uzlu je umožněno přijímat a odesílat procesní datové objekty (PDO). Uvnitř funkce `OperNode()`, která se cyklicky volá v operačním stavu uzlu, se postupně testují oba objekty zpráv Message 1 a Message 2. Testuje se zdali mají nastaven v řídících registrech Control 1 příznak NewDat, který indikuje příjem nových dat procesních datových objektů. V případě, že je tento příznak nastaven, je daný přijímací procesní objekt zpracován. To spočívá v přiřazení jednotlivých dat přenášených v tomto objektu příslušným proměnným odpovídajících aplikačních objektů. Toto přiřazení je závislé na mapovacích parametrech příslušných přijímacích PDO.

Po testování nových dat přijímacích procesních datových objektů a případné aktualizaci přijímacích aplikačních objektů je testována maticová klávesnice, zdali je stisknuta nějaká klávesa. Klávesnice je připojena k rozšiřujícímu portu P1 řadiče CAN, schéma zapojení je uvedeno na obr. 14. Dolní polovina portu je konfigurována jako vstupní a horní polovina jako výstupní.



obr. 14: Zapojení maticové klávesnice

Testování se provádí klasickým způsobem, kde se postupně na jeden ze čtyř sloupců zapíše hodnota log. 0 a na ostatní hodnota log. 1 a kontrolují se hodnoty logických úrovní na jednotlivých řádcích. Je-li některá z těchto úrovní log. 0, potom je stisknuta klávesa, která odpovídá řádku, ve kterém je hodnota log. 0 a sloupci na který byla zapsána hodnota log. 0. V případě, že je stisknuta klávesa, provede se podle kombinace logických úrovní na portu P1 její dekódování a tato hodnota se uloží do proměnné `Key`.

Pokud byla detekována stisknutá klávesa, což je podle aplikačního protokolu považováno za **událost**, jsou kontrolovány komunikační parametry obou **přijímacích** procesních datových objektů. Pokud má některý z nich nastaven typ přenosu na žádost (hodnota 02) a zároveň je v dodatečném kódu komunikačních parametrů uveden nižší bajt identifikátoru události vyvolané stiskem klávesy (hodnota 01), potom je příslušným přijímacím PDO **vyslán CAN rámec žádosti o data**. Dále jsou při této události vyvolané stiskem klávesy kontrolovány komunikační parametry **vysílacích** PDO, pokud je typ přenosu nastaven na událost a dodatečný kód komunikačních parametrů obsahuje nižší bajt identifikátoru události vyvolané stiskem klávesy (hodnota 01), potom je tento PDO **vyslán** na sběrnici CAN.

Dále je ve funkci zajišťující operační stav kontrolováno, zda je nastaven příznak `disp`, který se nastavuje každých 50 ms v obslužné rutině přerušení od časovače T0. Je-li tento příznak nastaven, aktualizuje se zobrazení LCD displeje podle konkrétních aplikačních objektů, které přísluší LCD displeji. Rovněž se vydá povel AD převodníku k převedení napětí z kanálu, který je uveden v proměnné `channel`, a takto získaná hodnota napětí je uložena do globální proměnné v poli `AD`, která odpovídá aplikačním objektům AD převodníku. Po každém převodu je hodnota proměnné `channel` inkrementována o jedničku a je kontrolováno, zda je rovna popřípadě větší než sedm. Pokud ano, je tato proměnná vynulována, aby se při příštím nastavení příznaku opět mohl převést první kanál AD převodníku. Tímto způsobem je každý kanál AD převodníku převáděn jednou za 400 ms (obnovovací frekvence 2,5 Hz). Tímto funkce operačního režimu končí a dokud není změněn stav uzlu, je volána opakovaně.

5.1.11 Konfigurační stav uzlu

Aby bylo možné konfigurovat parametry přenášených procesních datových objektů, je podle navrženého aplikačního protokolu umožněno uzel uvést do konfiguračního stavu. V tomto stavu se opakovaně volá funkce `PreopNode()`.

V těle této funkce se nejprve kontroluje proměnná `disp` zda obsahuje příznak „`DISP_CONF`“, který je nastavován vždy, když se uzel přepne do konfiguračního stavu. Když tento příznak obsahuje, tak se vypíše na první řádek LCD displeje textový řetězec

„Conf“ a následně je proměnná `disp` vynulována, protože při příštím volání této funkce by bylo zbytečné vypisovat stejný textový řetězec.

Dále se testuje jestli objekt zpráv Message 5 obsahuje nová data. Tento objekt zpráv je určen pro příjem příkazů servisních datových objektů. Detailní popis operací se servisními datovými objekty je popsána v kap. 4.4.5. Pokud byl přijat nový příkaz servisního datového objektu, je volána funkce `SDOServer()`, ve které je tento příkaz zpracován. Zpracování SDO příkazu spočívá v tom, že je tento příkaz rozkódován a dál se pokračuje podle toho, jestli se jedná o zápis mapovacích nebo komunikačních parametrů do objektového slovníku či o čtení těchto parametrů. V případě, že se nejedná ani o jeden z těchto dvou příkazů, je vyslána odpověď „chybný formát SDO příkazu“. Pokud se jedná o příkaz zápisu, jsou na příslušné místo v objektovém slovníku zapsána přijatá data. Proběhl-li zápis mapovacích nebo komunikačních parametrů korektně, jsou podle nich parametrizovány objekty zpráv procesních datových objektů, tyto nové parametry jsou rovněž zapsány do paměti EEPROM (konkrétní adresy jsou uvedeny v příloze) a je odeslána odpověď „zápis do objektového slovníku v pořádku“. Jestliže se při zápisu vyskytla chyba, je odeslána odpověď, ve která je chybový kód „zápis do objektového slovníku neproveden“. Pokud byl v přijatém rámci SDO příkazu detekován příkaz „čtení z objektového slovníku“, jsou přečteny hodnoty příslušného mapovacího nebo komunikačního parametru a tyto hodnoty jsou odeslány v SDO odpovědi. Jestliže při čtení z objektového slovníku došlo k chybě, je v SDO odpovědi odeslán chybový kód „chyba při čtení z objektového slovníku“.

Dále je ve funkci obsluhující konfigurační stav uzlu řešena obsluha případného datového toku prostřednictvím sériového kanálu, pokud dochází ke konfiguraci uzlu z osobního počítače. Podrobný popis sériové komunikace uzlu je popsán v kap. 6.3.

5.1.12 Stav nečinnosti uzlu

Dalším stavem, ve kterém se CAN uzel může nacházet je stav nečinnosti. V tomto stavu není možno přijímat ani odesílat procesní data a ani není umožněno uzel konfigurovat. V tomto stavu je opakovaně v modulu *main.c* volána funkce `PrepNode()`. V těle této funkce je programový kód, který se vykoná pouze při prvním volání této funkce po změně stavu uzlu do stavu nečinnosti.

Tento kód zajistí zobrazení textu „Stop“ na první řádek LCD displeje. Pokud byl uzel do tohoto stavu přepnut z podnětu při přerušení od CAN řadiče po výskytu chyby, je na druhý řádek LCD displeje vypsána zpráva „Err: CAN Warn“ nebo „Err: CAN BOff“, která charakterizuje chybu řadiče CAN. Příčiny výskytu těchto dvou chybových přerušení jsou uvedeny v kapitole 5.1.4. Po výskytu těchto chyb, je pro další použití CAN uzlu nutné provést reset CAN uzlu odpojením a připojením napájecího napětí.

5.1.13 Obsluha přerušení od řadiče CAN

Přerušovací výstup $\overline{\text{INT}}$ řadiče CAN je připojen na vstup $\overline{\text{INT0}}$ mikrořadiče AT89C52 a slouží tedy jako zdroj externího přerušení. V softwarové obsluze tohoto přerušení se podle obsahu registru přerušení (Interrupt Register) v paměti RAM CAN řadiče rozliší zdroj přerušení.

Obsahuje-li Interrupt Register hodnotu 1, je zdrojem přerušení výskyt chyby CAN řadiče (status Warning nebo Bus Off). V takovémto případě se do globální proměnné `State` zapíše taková hodnota, kterou se CAN uzel uvede do stavu nečinnosti a do proměnné `disp` se nastaví příznak, který zajistí zobrazení chybové zprávy na LCD displeji.

Jestliže Interrupt Register obsahuje hodnotu 10, přerušení bylo vyvoláno přijetím synchronizačního objektu (SYNC viz kap. 4.4.7). Je vynulován příznak IntPnd v řídicím registru Control 0 objektu zpráv Message 8. Reakcí na přijetí synchronizačního objektu je vyslání všech vysílacích procesních datových objektů, v jejichž komunikačních parametrech je nastaven typ přenosu na hodnotu 1, což znamená **synchronní přenos**.

Pokud obsahuje Interrupt Register hodnotu 9, bylo přerušení vyvoláno v závislostí na přijetí NMT rámce (viz kap. 4.4.6), pomocí kterého lze přepínat mezi stavy CAN uzlu. Jako v předchozím případě je nejprve vynulován příznak IntPnd objektu Message 7. NMT příkaz je zpracován, což spočívá v porovnání prvního bajtu tohoto příkazu s identifikátorem CAN uzlu (hodnota identifikátoru CAN uzlu: 10). Pokud tyto hodnoty nesouhlasí, pak přijatý příkaz není určen pro tento uzel a obsluha přerušení končí. V opačném případě je tento příkaz dále zpracován. Při tomto dalším

zpracování je podle obsahu druhého bajtu rozhodnuto, do kterého následujícího stavu uzel přepnut.

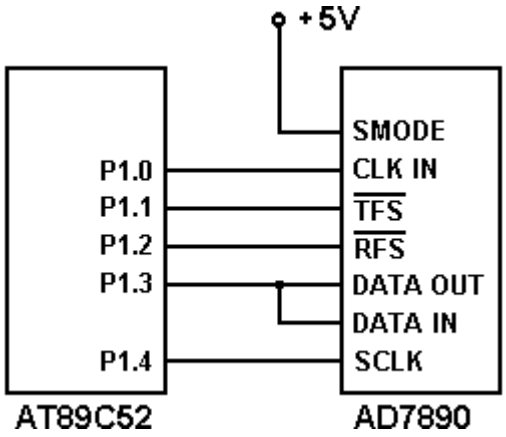
Při přepínání do konfiguračního stavu, jsou deaktivovány první čtyři objekty zpráv (Message 1 až Message 3), které jsou určeny pro příjem a vysílání procesních datových objektů. Deaktivace objektů zpráv se provádí tak, že je vynulován příznak MsgVal v registru Control 0 příslušného objektu zpráv. Dále jsou aktivovány (validovány) objekty zpráv Message 5 a Message 6 pro příjem a odesílání servisních datových objektů. Dále je deaktivován objekt zpráv Message 8 pro příjem synchronizačního objektu. A nakonec je do globální proměnné `disp` zapsán příznak, který zajistí vypsání textového řetězce „Conf“ na prvním řádku LCD displeje.

Pokud má být uzel přepnut do operačního stavu, jsou jednotlivé objekty zpráv aktivovány a deaktivovány přesně opačně, než tomu bylo v předchozím případě při přepnutí uzlu do konfiguračního stavu. Objekty zpráv procesních datových objektů jsou aktivovány, objekty pro přenos SDO rámců jsou deaktivovány a objekt zpráv pro příjem synchronizačního rámce je aktivován. Nakonec je pomocí proměnné `disp` zajištěno, aby se na LCD displeji zobrazovaly přijímané aplikační objekty LCD displeje.

V případě požadavku zastavení uzlu, jsou všechny objekty zpráv pro PDO, SDO a synchronizační objekt deaktivovány. A do globální proměnné `disp` je zapsán příznak, který zajistí, že se na prvním řádku LCD displeje zobrazí nápis „Stop“.

5.1.14 Operace s AD převodníkem AD7890

Převodník je k mikrořadiči připojen podle schématu na obr. 15. Vstup SMODE je připojen k napětí s úrovní log. 1. To znamená, že převodník je provozován v módu externího časování, kde je sériový přenos dat časován signálem SCLK.



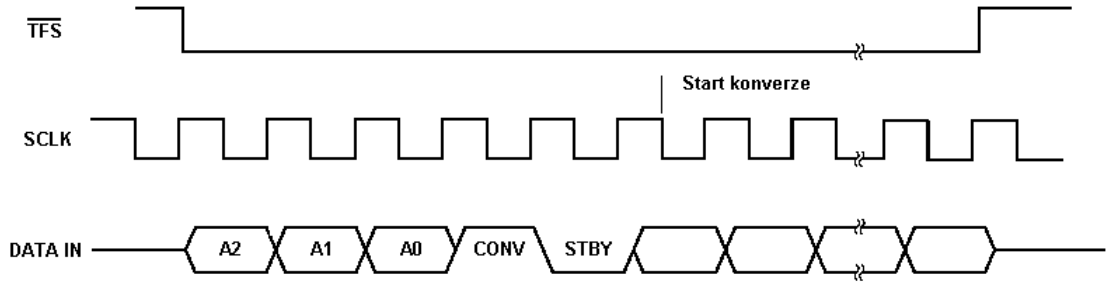
obr. 15: Propojení mezi AT89C52 a AD7890

Pokud je třeba převést vstupní napětí libovolného kanálu AD převodníku, je nutné do řídicího registru zapsat příkaz, který obsahuje adresu (bity A0 až A2) jednoho z osmi vstupů. Nastavením bitu CONV se vydá povel ke spuštění konverze napětí z odpovídajícího kanálu. Řídicí registr ještě obsahuje pátý bit STBY, kterým je možné převodník uvést do stavu se sníženou spotřebou, tento režim ale v řídicím programu není využit.

4	3	2	1	0
A2	AI	A0	CONV	STBY

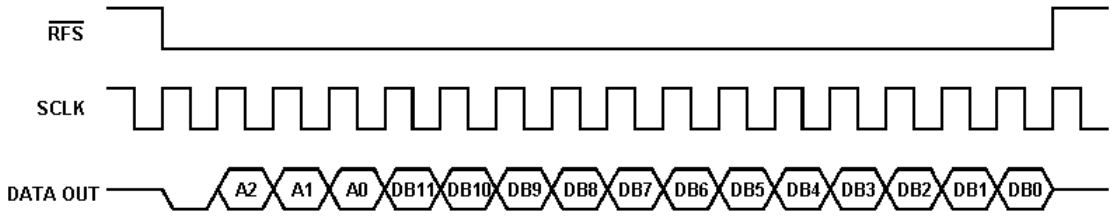
obr. 16: Řídicí registr AD převodníku AD7890

Zápis do řídicího registru začíná nastavením signálu $\overline{\text{TFS}}$ na úroveň log. 0. Potom se na vstup DATA IN začne přenášet potřebný příkaz, který je potvrzován hodinovým signálem SCLK. Po přenesení všech pěti bitů řídicího registru a následné sestupné hraně signálu SCLK se spustí konverze odpovídajícího kanálu převodníku. Na případných dalších zapisovaných bitech po přenesení řídicího slova nezáleží. Zápis dat do převodníku končí v okamžiku, kdy je nastaven signál $\overline{\text{TFS}}$ do log. 1. Zápis příkazu ke konverzi do řídicího registru převodníku je znázorněn na následujícím obrázku.



obr. 17: Průběh signálů při zápisu do AD převodníku

Čtení binárních hodnot převedeného napětí musí začít vynulováním signálu $\overline{\text{RFS}}$. Potom následuje příjem šestnáctibitové zprávy, která začíná nulovým bitem a za ním následují tři bity, které obsahují číslo naposledy převáděného kanálu a dvanáct bitů s binární hodnotou převedeného napětí. Čtení dat končí nastavením signálu $\overline{\text{RFS}}$. Vše je názorně vidět na obr. 18.



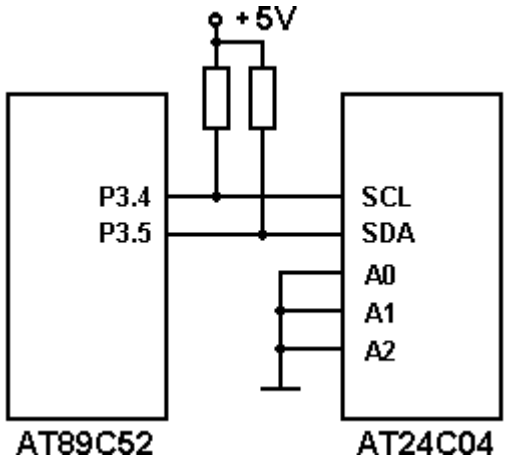
obr. 18: Průběhy signálů při čtení z AD převodníku

Funkce pro komunikaci s AD převodníkem jsou naprogramovány ve zdrojovém souboru *adc.c*. Více podrobností o tomto převodníku lze nalézt v [9].

5.1.15 Operace s pamětí EEPROM AT24C04

Tato napěťově nezávislá paměť s kapacitou 512 B je určena pro ukládání mapovacích a komunikačních parametrů procesních datových objektů. Při opětovném spuštění uzlu jsou v případě platnosti dat podle těchto parametrů nastaveny procesní datové objekty (viz kap. 4.4.1). Rozmístění konkrétních položek v paměti je uvedeno v příloze.

Komunikace s EEPROM probíhá po rozhraní I^2C, které je tvořeno signály SDA a SCL.



obr. 19: Propojení mezi AT89C52 a AT24C04

Operace zápisu dat do paměti vždy začíná startbitem (sestupná hrana signálu SDA při nastaveném signálu SCL). Za ním následuje jeden bajt, který obsahuje adresu obvodu a bit, podle kterého se data zapisují nebo čtou. Tento bajt má binární hodnotu 1010000. První čtyři bity jsou součástí pevné adresy obvodu AT24C04, další dva bity odpovídají zapojení adresových vstupů A1 a A2 obvodu, které jsou připojeny na úroveň log. 0. Dalším bitem je vybrána dolní stránka paměti o velikosti 256 B. Poslední nulový bit symbolizuje operaci zápisu do paměti. Po odeslání tohoto bajtu vyšle obvod AT24C04 potvrzení o přijetí (nastaví SDA do log. 0). Dále je třeba paměti předat adresu o délce osmi bitů, na kterou se bude zapisovat, po přijetí obvod opět vyšle potvrzení a je možné zapisovat data o délce osmi bitů. Po každém přijatém bajtu dojde k uložení a k potvrzení. Každý další přijatý bajt je uložen na adresu zvýšenou o jedničku než předchozí. Zápis dat do paměti končí stopbitem (náběžná hrana signálu SDA při nastaveném signálu SCL).

Při čtení z paměti je nutné vygenerovat startbit, vyslat bajt s adresou obvodu (hodnota 10100001) a další bajt s adresou, ze které se má číst. Potom se musí vygenerovat nový startbit, za ním opět bajt s adresou obvodu a až potom bude paměť vysílat přečtená data. Každý přijatý bajt z paměti je nutné potvrdit (vynulovat SDA), tímto potvrzením je inkrementována adresa v paměti, ze které se přečte další bajt. Poslední přečtený bajt není nutné potvrdit a pak následuje stopbit.

Funkce pro komunikaci mikrořadiče s pamětí AT24C04 jsou naprogramované v modulu *eeprom.c*.

5.2 Modul na bázi C167CR

Navržený aplikační protokol byl rovněž implementován do mikropočítačových modulů s mikrořadičem Infineon C167CR. Stejně jako při implementaci do CAN uzlu s mikrořadičem AT89C52, byl řídicí program napsán v jazyce C ve vývojovém prostředí Keil µVision2.

Je umožněno přijímat dva a odesílat dva procesní datové objekty (viz kap. 4.4.1). Modul je vybaven jedním SDO serverem, pomocí kterého je možné konfigurovat jeho mapovací a komunikační parametry a jedním SDO klientem, díky kterému je umožněno přistupovat k parametrům okolních jednotek a modifikovat tak jejich přenosové vlastnosti. Dále tyto jednotky na bázi C167CR mohou vysílat NMT příkazy a tak přepínat stavy okolních uzlů.

Řídicí program umožňuje, aby modul s C167CR byl provozován jako řídicí jednotka (Master) CAN sítě nebo jako jednotka podřízená (Slave). Jestli má být mikropočítačový modul provozován jako Master nebo jako Slave záleží na tom, zda byl řídicí program zkompilován s definicí `MASTER` v hlavičkovém souboru *param.h*. Tato definice má formát:

```
#define MASTER
```

Pokud jsou zdrojové soubory zkompilovány s výše uvedenou definicí, pak bude modul plnit funkci řídicí jednotky Master. To znamená, že bude na sběrnici vysílat synchronizační objekty (SYNC) v pravidelných časových intervalech s periodou 100 ms. V případě, že má být modul provozován v režimu podřízené jednotky (Slave) musí být výše uvedená definice vynechána.

Struktura řídicího programu pro modul s mikrořadičem C167CR je podobná struktuře řídicího programu CAN uzlu s mikrořadičem AT89C52.

5.2.1 Uživatelské rozhraní

Modul na bázi C167 má v implementaci aplikačního protokolu definováno šest aplikačních objektů (viz kap. 4.3.3), které odpovídají vybraným vstupům. A čtyři výstupní aplikační objekty. Přiřazení jednotlivých aplikačních objektů je patrné z následující tabulky.

	Název	Index [hex]	Subindex [hex]	Velikost [b]
Vstupy	Analog.vstup AIN0	0700	01	16
	Analog.vstup AIN1	0700	02	16
	Analog.vstup AIN2	0700	03	16
	Analog.vstup AIN3	0700	04	16
	Analog.vstup AIN4	0700	05	16
	Kód klávesy	0710	00	8
Výstupy	LED 1	0800	00	8
	LED 2	0801	00	8
	LCD 1	0810	00	8
	LCD 2	0811	00	8

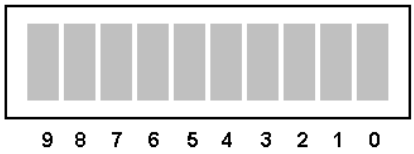
tab. 14: Umístění aplikačních objektů v objektovém slovníku

Vstupy modulu:

- *Analogové vstupy* (AIN0 až AIN5): jsou vstupy analogově-digitálního převodníku, které mají v oblasti aplikačních objektů přiřazen jednotný index 0700 hex a subindex podle příslušného vstupu. Na tyto analogové vstupy je připojeno pět potenciometrů.
- *Kód klávesy:* tento aplikační objekt obsahuje osmibitový binární kód naposledy stisknuté klávesy.

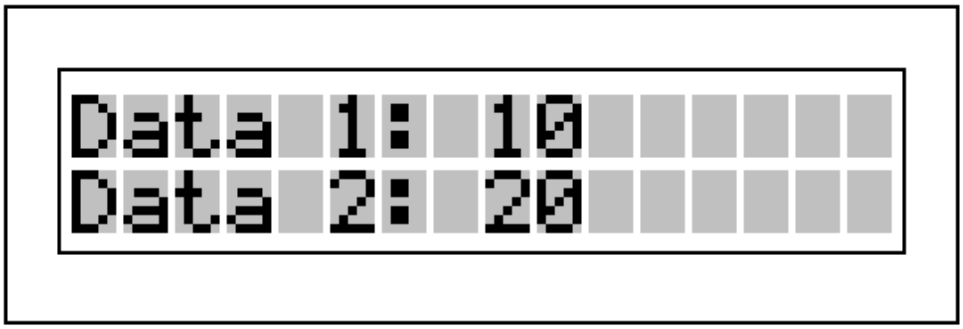
Výstupy modulu:

- *LED 1:* tento aplikační objekt o délce osmi bitů představuje dolních osm z deseti LED diod na desce modulu. Nejméně významnému bitu odpovídá LED dioda na obr. 20: označená nulou a nejvíce významnému odpovídá LED 7. Nastavením bitu se příslušná dioda rozsvítí.



obr. 20: Označení jednotlivých LED diod

- *LED 2:* modifikací tohoto aplikačního objektu lze ovládat zbylé dvě z deseti LED diod. Nultému bitu odpovídá LED 8 a prvnímu bitu dioda LED 9. Na zbylých šesti diodách nezáleží.
- *LCD 1:* tento aplikační objekt má délku jednoho bajtu a zapsáním dat do něj se zobrazí na prvním řádku LCD displeje příslušné hexadecimální číslo (0 až FF). Zobrazení displeje v operačním stavu uzlu je vyobrazeno na následujícím obrázku.
- *LCD 2:* aplikační objekt podobný jako v předchozím případě, ale s tím rozdílem, že zápisem do tohoto objektu bude hexadecimální číslo zobrazeno na druhém řádku LCD displeje.



obr. 21: vzhled LCD displeje v operačním stavu uzlu

5.2.2 Programování modulu

Zavedení zkompilovaného řídicího programu do mikrořadiče C167CR se provádí pomocí osobního počítače v prostředí Hitop dodávané firmou Hitex Ltd. jako součást tohoto kitu.

Program mikrořadiče C167CR se skládá ze zdrojových a hlavičkových souborů:

start167.a66
main.c
can_a.c
can_regs.c
lcd.c
led.c
keypad.c

Zdrojový soubor *start167.a66* napsaný v asembleru je součástí vývojového prostředí. Jeho kód obsahuje inicializační sekvenci mikrořadiče C167CR, která se vykoná bezprostředně po resetu. Modul *main.c* obsahuje volání inicializační funkce a uzavřenou programovou smyčku. V modulu *can_regs.c* a jeho hlavičkovém souboru jsou obsaženy definice jmen registrů CAN řadiče. Zdrojový soubor *can_a.c* obsahuje hlavní funkce aplikačního protokolu. V ostatních zdrojových souborech jsou funkce pro LCD displej, LED diody a klávesnici. Všechny zdrojové texty jsou na CD v příloze.

5.2.3 Nastavování parametrů aplikačního protokolu

Hlavičkový soubor *param.h*, který je součástí zdrojových souborů obsahuje definice, na kterých závisí vlastnosti aplikačního protokolu a tím chování uzlu.

Jako první lze v tomto souboru nastavit identifikátor modulu (viz kap. 4.2). Dále se pomocí definice „Master“ stanoví, zda jednotka bude plnit řídicí funkci na sběrnici CAN. Vynecháním této definice bude jednotka v režimu „Slave“.

Jako další se v tomto hlavičkovém souboru nastavují komunikační a mapovací parametry všech čtyř procesních datových objektů. Způsob nastavování jednotlivých parametrů spočívá v přiřazení konkrétních hodnot symbolickým jménům, které představují jednotlivé parametry. Obsah hlavičkového souboru *param.h* a postup nastavování parametrů je uveden v příloze.

5.2.4 Vysílání rámců s NMT příkazy

Pomocí těchto příkazů lze řídit provozní stavy okolních uzlů (viz kap. 4.4.6). Pro tento účel je v modulu *can_a.c* naprogramovaná funkce `NMT`, která zajistí vyslání rámce na sběrnici CAN.

```
void NMT(unsigned char cmd, unsigned char id)
```

Tato funkce má dva parametry. Do parametru `cmd` se vkládá kód, který charakterizuje NMT příkaz a parametr `id` musí obsahovat identifikátor modulu, pro který je příkaz určen. Zakódované hodnoty příkazu jsou uvedeny v tab. 10 v kap. 4.4.6. Místo konkrétních hodnot kódu příkazu je možné funkci volat s předdefinovanými konstantami:

```
NMTCMD_CONF
NMTCMD_START
NMTCMD_STOP
```

Hodnoty těchto konstant odpovídají kódům příkazu pro stav konfigurace, operační stav a pro příkaz k zastavení jednotky.

Tato funkce je připravena, tak aby bylo umožněno jí vložit do řídicího programu.

Konkrétně v přiloženém programu pro C167 je tato funkce s různými parametry volána uvnitř těla funkce `OperNode()` jako reakce na stisk klávesy „1“, „2“ nebo „3“. Všechny tři příkazy jsou určeny jednotce s hodnotou identifikátoru modulu 10, tato hodnota odpovídá identifikátoru CAN uzlu. Při stisku klávesy „1“ je umožněno přepnou CAN uzel do operačního režimu. Stiskem „2“ se CAN uzel zastaví a po stisknutí „3“ dojde k přepnutí do konfiguračního stavu.

5.2.5 Konfigurace vzdáleného uzlu.

Jak již bylo zmíněno, je modulům s C167CR umožněno vysílat příkazy pro zápis nebo čtení parametrů okolních uzlů (SDO client viz kap. 4.4.5). Ve zdrojovém souboru *can_a.c* je připravena funkce `SendSDO`, která odešle SDO příkaz a čeká na obdržení odpovědi.

V parametrech tohoto příkazu je kód charakterizující zápis nebo čtení do/z objektového slovníku. Index a subindex zapisované nebo čtené položky, pole čtyř proměnných o velikosti 1 B pro předání dat. Jedná-li se o zápis parametrů do objektového slovníku, jsou zapsány hodnoty předané v tomto poli.

Funkce navrací hodnotu, v závislosti na úspěšnosti operace. V hlavičkovém souboru *can_a.h* jsou definovány konstanty možných návratových hodnot.

Syntaxe příkazu je následující:

```
char SendSDO( char cmd,
              char id,
              int Index,
              char SubIndex,
              char *d)
```

Parametry:

cmd:	zápis do obj. slovníku:	hodnota 10 hex nebo konst. SDOCMD_WR
	čtení z obj. slovníku	hodnota 20 hex nebo konst. SDOCMD_RD
Index, Subindex:		hodnota indexu a subindexu položky
d:		pole zapisovaných nebo přečtených dat

Návratové hodnoty:

SDOCMD_WROK:	zápis v pořádku.
SDOCMD_WRERR:	chyba při zápisu.
SDOCMD_RDOK:	čtení v pořádku, potom pole d obsahuje přečtená data.
SDOCMD_RDERR:	chyba při čtení
SDOCMD_ERR:	odpověď na neznámý SDO příkaz.

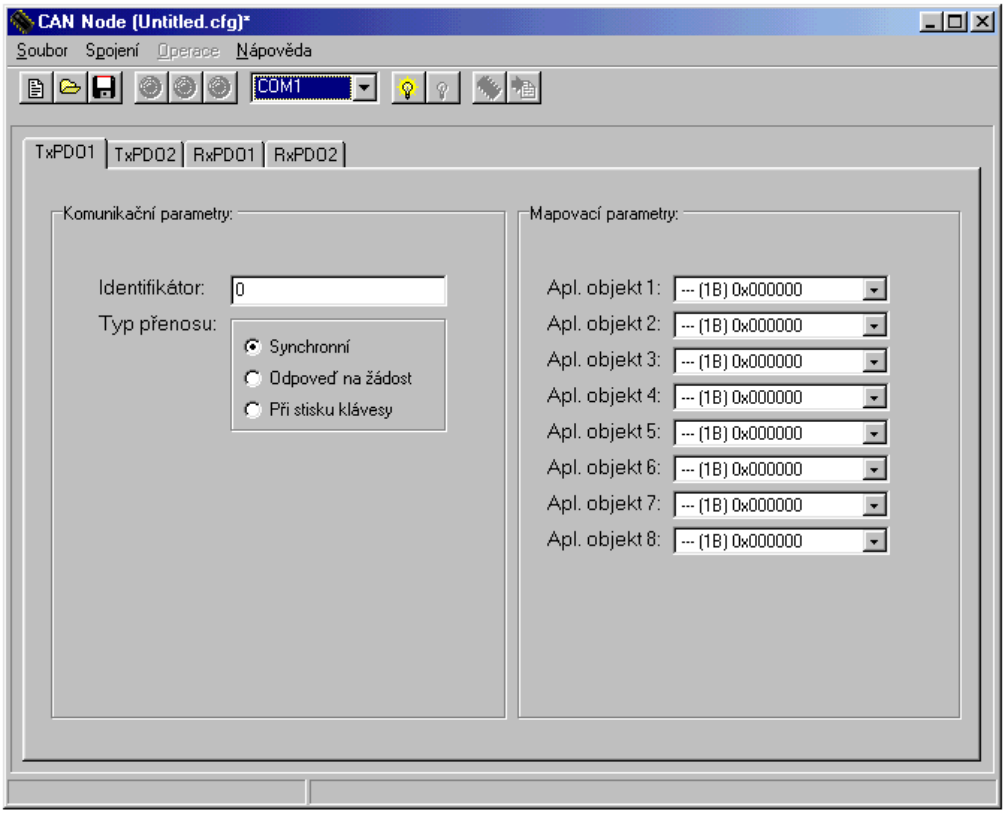
Použití této funkce je názorné v přiloženém programu pro C167. Funkce `SendSDO` je volána při stisku kláves „8“ nebo „9“. Stiskem klávesy „8“ dojde k zápisu do mapovacích parametrů. Po stisku klávesy „9“ je přečten jeden záznam komunikačních parametrů a přečtené hodnoty jsou zobrazeny na LCD displeji.

6 Konfigurační aplikace pro CAN uzel

6.1 Popis aplikace

Tato aplikace slouží pro konfiguraci parametrů implementovaného aplikačního protokolu v CAN uzlu s mikrořadičem AT89C52 přes sériové rozhraní. Je naprogramována ve vývojovém prostředí Borland Delphi 5 od společnosti Inprise Corporation a je určena pro operační systémy Microsoft Windows 95/98/ME/NT/2000/XP. Zdrojové soubory lze nalézt na přiloženém CD.

Pomocí této aplikace je možné přepínat mezi operačním a konfiguračním stavem nebo je možné uzel zastavit. V konfiguračním stavu je umožněno z uzlu načítat nebo do něho zapisovat konkrétní hodnoty komunikačních a mapovacích parametrů procesních datových objektů (viz kap. 4.3.1).



obr. 22: Hlavní okno konfigurační aplikace

Uživatelské rozhraní této aplikace je navrženo v českém jazyce. Ovládání je intuitivní v souladu s obvyklými postupy. Pro větší uživatelský komfort je většina položek z hlavního menu umístěná ve formě tlačítek se zobrazenými grafickými

symboly na nástrojové liště. Umístěním kurzoru myši nad tlačítko na nástrojové liště je zobrazena „plovoucí nápověda“ s popisem funkce příslušného tlačítka.

Na formuláři aplikace jsou umístěny čtyři záložky. Dvě záložky jsou určeny pro vysílací procesní datové objekty (ozn. TxPDO1 a TxPDO2) a zbylé dvě záložky jsou určeny pro přijímací datové objekty (ozn. RxPDO1 a RxPDO2). Na každé ze záložek jsou editační prvky pro komunikační a mapovací parametry příslušného procesního datového objektu podle aktivní záložky.

Aplikace umožňuje parametry na všech záložkách formuláře uložit na pevný disk jako soubor s konfigurací nebo je z již uloženého konfiguračního souboru načíst. Tento konfigurační soubor s příponou cfg je uložen v textové formě s přesně definovaným umístěním jednotlivých parametrů. Struktura konfiguračního souboru je uvedena v příloze.

6.2 Použití aplikace

Před vlastními operacemi s CAN uzlem je třeba s ním navázat spojení přes příslušný sériový port. To lze provést pomocí položky „Spojení“ hlavního menu nebo výběrem příslušného portu z nabídky na nástrojovém panelu a kliknutím na tlačítko se symbolem rozsvícené žárovky. Ukončit toto spojení lze rovněž z nabídky „Spojení“ hlavního menu nebo kliknutím na tlačítko se symbolem zhasnuté žárovky.

Pokud je otevřen příslušný sériový port, je zpřístupněna položka „Operace“ hlavního menu a tři tlačítka na nástrojové liště se symboly červené, žluté a zelené kontrolky. Pomocí této nabídky a těchto tlačítek lze přepínat mezi stavy CAN uzlu. Aktuální stav je zobrazován ve stavovém řádku aplikace.

K tomu, aby bylo možné z uzlu načíst nebo do něho uložit parametry procesních datových objektů je nutné jej přepnout do stavu konfigurace. Tím jsou zpřístupněny zbylé dvě položky v hlavním menu a poslední dvě tlačítka na nástrojové liště. Jejich význam je načtení nebo uložení konfigurace z/do CAN uzlu.

6.2.1 Načítání parametrů

Načtením parametrů z CAN uzlu dojde k automatické modifikaci editačních prvků na všech záložkách v závislosti na konkrétních komunikačních a mapovacích

parametrech. Hodnoty zobrazené na těchto prvcích potom odpovídají načteným parametrům.

6.2.2 Zápis parametrů

Pokud je vydán povel k zapsání konfigurace do CAN uzlu buď z hlavní nabídky v položce „Operace“ anebo kliknutím na tlačítko na nástrojové liště, je nutné, aby uživatel potvrdil potvrzovací zprávu. Nedojde-li k potvrzení této zprávy, žádné nové parametry se do uzlu neuloží. V případě, že je tato zpráva potvrzena, jsou do CAN uzlu uloženy takové parametry, které odpovídají hodnotám na editačních prvcích všech záložek.

Uživatelské rozhraní umožňuje do uzlu uložit jen takové parametry, které odpovídají implementovanému aplikačnímu protokolu v CAN uzlu.

Identifikátory vysílacích i přijímacích procesních datových objektů (PDO) mohou být podle navrženého protokolu buď nulové, a to v případě, že daný PDO není použit k přenosu nebo mohou mít hodnoty v rozmezí 384 až 1023.

Další komunikační parametry vysílacích procesních datových objektů je možné nastavit tak, že daný PDO může být vysílán synchronně, jako odpověď na žádost jiného uzlu nebo může být vyslán při stisknutí klávesy (což je v apl. protokolu definováno, jako vyslání, v závislosti na výskytu události). Pro přijímací PDO je možné nastavit do příslušného komunikačního parametru typ přenosu synchronní nebo je možné nastavit, že tento PDO vyšle rámec žádosti o data při stisknutí klávesy.

Do mapovacích parametrů vysílacích a přijímacích PDO lze pomocí této aplikace nastavit jen takové aplikační objekty, které má CAN uzel definované v aplikačním protokolu (viz kap. 5.1.2).

6.3 Sériová komunikace

Komunikace mezi počítačem a CAN uzlem přes sériový port je naprogramována s využitím API služeb jádra operačního systému Microsoft Windows. Komunikace probíhá rychlostí 9600 b/s. Jeden rámec obsahuje startbit, osm datových bitů a jeden stopbit. Parita není pro přenos využita.

6.3.1 Příkazy ke změně stavu uzlu

Změna stavu CAN uzlu přes sériovou linku spočívá ve vyslání jednobajtového příkazu. Pokud uzel tento příkaz přijme, je zpracován a odeslána jednobajtová odpověď. Pokud korektně došlo ke změně stavu, je odeslána příslušná odpověď. V případě, že přijatý příkaz nebyl zpracován není odeslána žádná odpověď. Pokud konfigurační aplikace obdrží jinou než předpokládanou odpověď nebo do dvou sekund neobdrží žádnou odpověď, je to klasifikováno jako chyba a je zobrazeno chybové hlášení. Konkrétní hodnoty příkazů a odpovědí jsou uvedeny v tab. 15.

Stav	Příkaz [hex]	Odpověď [hex]
Operační	A5	AA
Uzel zastaven	B5	BA
Konfigurační	C5	CA

tab. 15: Hodnoty příkazů a odpovědí pro řízení stavu CAN uzlu

6.3.2 Zapisování konfigurace

Při zapisování konfigurace jsou do CAN uzlu postupně vyslány komunikační a mapovací parametry všech PDO. Zapisované hodnoty odpovídají hodnotám editačních prvků na všech záložkách aplikace.

Vlastní přenos dat probíhá podle jednoduchého echo protokolu, kde se z PC vyšle jeden bajt, mikrořadič ho přijme a odešle zpět. Přijatý bajt je porovnán s odvysílaným a pokud jsou shodné, může být odvysílán další.

Zapisovací sekvence začíná vysláním jednobajtového příkazu s hodnotou 10 hex, který charakterizuje zápis parametrů. Za ním následují dva bajty indexu a jeden bajt subindexu konkrétního umístění v objektovém slovníku, kam se bude zapisovat. Nakonec jsou přenášeny čtyři bajty odpovídající hodnotám příslušných parametrů. Po přenesení této sekvence dojde zapsání konkrétních hodnot do objektového slovníku mikrořadiče. Pokud zápis proběhl v pořádku, je odeslána odpověď, kterou představuje hodnota 51 hex. Jestliže se při zápisu do objektového slovníku vyskytla chyba, je jako odpověď odeslána hodnota 52 hex a aplikace zobrazí chybové hlášení.

6.3.3 Načítání konfigurace

Postupně jsou z paměti mikrořadiče načteny parametry všech komunikačních i mapovacích parametrů procesních datových objektů.

Proces načítání spočívá v odeslání sekvence, která obsahuje jednobajtový příkaz (hodnota 20 hex), dva bajty obsahující index a jeden bajt se subindexem. V závislosti na přijetí této sekvence mikrořadič provede čtení parametrů objektového slovníku z konkrétního umístění. Pokud bylo čtení úspěšné, je odeslána hodnota 61 hex a za ní následují čtyři bajty s přečtenými daty. Pokud však čtení komunikačních nebo mapovacích parametrů nebylo úspěšné, mikrořadič odešle chybový kód 62 hex a konfigurační aplikace nahlásí chybové hlášení.

7 Závěr

7.1 Možnosti aplikačního protokolu

Navržený aplikační protokol pro sběrnici CAN dovoluje přenášet data mezi periferiemi jednotlivých modulů v síti. Přenosová rychlost je 100 kb/s. Datové přenosy jsou umožněny v předdefinovaných formách třemi různými způsoby. Z důvodů možnosti konfigurace parametrů zařízení a díky předem stanoveným aplikačním objektům se zařízení s tímto protokolem stává velice flexibilním.

Mezi hlavní výhody tohoto aplikačního protokolu lze počítat například obecnost návrhu, nízké nároky na hardware, flexibilitu zařízení a snadnou implementaci.

Nevýhodami aplikačního protokolu jsou například pevně stanovená komunikační rychlost, absence detekování chybových stavů zařízení a omezený počet přenosových způsobů.

7.2 Způsob implementace

Řídicí program pro CAN uzel s mikrořadičem Atmel AT89C52 byl vytvořen ve vývojovém prostředí Keil µVision2. Zdrojové texty byly napsány v jazyce C, což je v souladu se zadáním. Programová paměť AT89C52 byla naprogramována s použitím univerzálního programátoru ATmega od firmy MITE Hradec Králové, s.r.o.

Pro snadnou konfiguraci a kontrolu mapovacích a komunikačních parametrů byla vytvořena konfigurační aplikace pro PC. Tato aplikace komunikuje s CAN uzlem prostřednictvím sériového kanálu.

Rovněž program pro mikropočítačový systémy s mikrořadičem Infineon C167CR byl vytvořen ve vývojovém prostředí Keil µVision2 v jazyce C. Zkompilovaný kód byl do modulů zaveden pomocí osobního počítače přes sériový kanál pomocí aplikace Hitop od společnosti Hitex Ltd, která dodává tyto mikropočítačové moduly.

7.3 Ověření funkčnosti

Funkčnost řídicích programů s implementací navrženého aplikačního protokolu byla ověřena jak u CAN uzlu s mikrořadičem Atmel AT89C52, tak i u modulu na bázi Infineon C167CR. Toto ověření bylo provedeno v mikropočítačové laboratoři na Katedře softwarového inženýrství.

K tomuto účelu byl CAN uzel na bázi 8052 propojen sběrnicí CAN s jedním modulem s C167CR. Systém s mikrořadičem C167CR zastával funkci řídicí jednotky (Master) a CAN uzel k němu byl připojen jako jednotka podřízená (Slave).

U obou jednotek byly postupně testovány veškeré jejich vlastnosti z hlediska aplikačního protokolu. Byla odzkoušena komunikace mezi periferiemi obou jednotek pomocí přenosu v procesních datových objektech (PDO) a přenos rámců s příkazy řídicími stavy uzlu. Dále pak konfigurace parametrů zařízení s využitím přenosu servisních datových objektů.

Veškerá komunikace po sběrnici CAN probíhala bez problémů. Všechna data mezi periferiemi obou modulů byla předávána přesně tak, jak bylo definováno v mapovacích a komunikačních parametrech příslušného zařízení.

Bez problémů fungovalo i vzdálené ovládání uzlu přes sběrnici CAN pomocí rámců s NMT příkazy.

Byl testován i přenos rámců se servisními datovými objekty funkce. Při operacích čtení mapovacích a komunikačních parametrů z objektového slovníku CAN uzlu byly obdrženy odpovídající odpovědi. Zápis mapovacích i komunikačních parametrů do CAN uzlu pomocí SDO byl rovněž úspěšný. To bylo ověřeno následným přenosem procesních datových objektů mezi periferiemi jednotkami, který probíhal podle nových parametrů. Ověření rovněž spočívalo v načtení parametrů pomocí konfigurační aplikace přes sériové rozhraní do osobního počítače. Takto načtené hodnoty odpovídaly parametrům, které byly do uzlu zaslány.

7.4 Možnosti zdokonalení

V případném dalším vývoji by bylo vhodné tento aplikační protokol například doplnit o objekty, které by kontrolovaly chybovost a detekovaly případné provozní výpadky jednotlivých připojených uzlů. Potom by tento protokol mohl být i použit pro řízení v reálných aplikacích. Dále by bylo například vhodné umožnit výběr z více

přenosových rychlostí po sběrnici atd. Možnosti dalších zdokonalení a úprav jsou téměř nevyčerpatelné a závisejí na případném konkrétním odvětví použití.

Literatura

[1] Novák, L.: Realizace uzlu s protokolem CAN, Diplomová práce, TU v Liberci 2003

[2] CAN in Automation e. V.: CANopen Communication Profile for Industrial Systems, CiA DS 301 V 3.0

[3] Robert Bosch GmbH: CAN Specification Version 2.0, Robert Bosch GmbH 1991

[4] Atmel Corporation: 8-bit Microcontroller with 8K Bytes Flash AT89C52, Atmel Corporation 1999

[5] Intel Corporation: 82527 Serial Communications Controller Architectural Overview, Intel Corporation 1996

[6] Intel Corporation: 82527 Serial Communications Controller, Controller Area Network Protocol, Intel Corporation 1995

[7] Intel Corporation: Interfacing an MCS® 51 Microcontroller to an 82527 CAN Controller, Intel Corporation 1996

[8] Atmel Corporation: 2-Wire Serial EEPROM AT24C01A/02/04/08/16, Atmel Corporation

[9] Analog Devices:LC^{2}MOS8-Channel, 12-Bit Serial, Data Acquisition System, Analog Devices, Inc., 2001

[10] Infineon Technologies:C167CR Derivates, 16-Bit Single-Chip Microcontroller, Infineon Technologies, AG., 2000

[11] Skalický, P.: Mikroprocesory řady 8051, BEN – technická literatura, Praha 2002

Přílohy

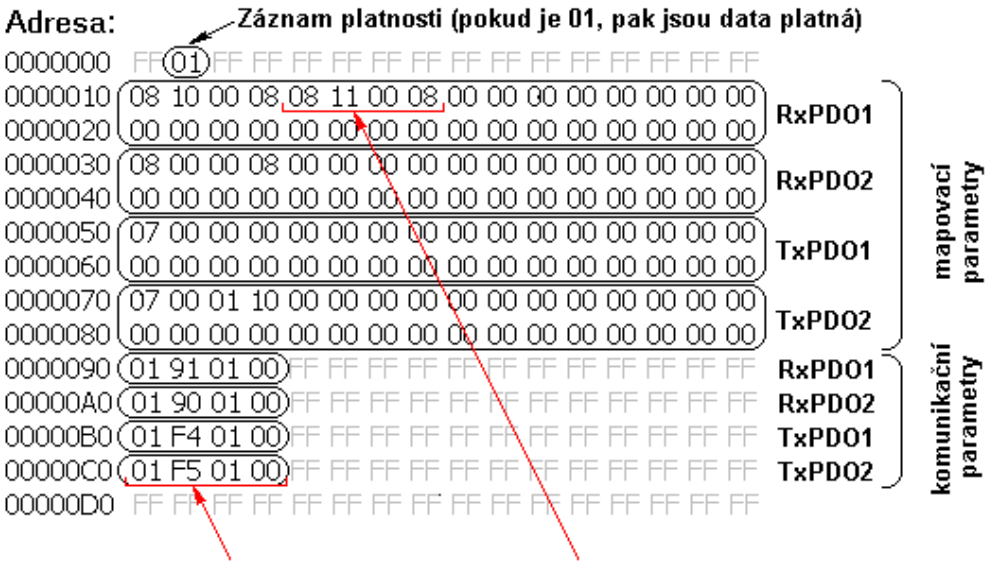
Rozmístění položek v paměti EEPROM AT24C04

Struktura souboru konfigurační aplikace s uloženými parametry (*.cfg)

Postup parametrizace modulů C167 pomocí souboru param.h

Software na CD ROM

Rozmístění položek v paměti EEPROM AT24C04



Struktura souboru konfigurační aplikace s uloženými parametry (.cfg)*

Soubor se skládá ze záhlaví a jednotlivých mapovacích a komunikačních parametrů všech čtyř procesních datových objektů ve formě textu. Mapovací parametr jednotlivých PDO obsahuje pole osmi struktur mapovacích parametrů a jednu strukturu komunikačních parametrů.

```
[CAN Node Conf]
[DATE]
 19.5.2004
[TIME]
 13:01:05
[RxPDO1-map]
 2064 0 8
 2065 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
[RxPDO1-comm]
 401 1 0
[RxPDO2-map]
 2048 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
[RxPDO2-comm]
 400 1 0
[TxPDO1-map]
 1792 0 16
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
[TxPDO1-comm]
 500 1 0
[TxPDO2-map]
 1792 1 16
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
 0 0 8
[TxPDO2-comm]
 501 1 0
```

Struktura mapovacích parametrů:
(záznam o délce 4 B)

- 2 B Index apl. objektu
- 1 B Subindex apl. objektu
- 1 B Délka apl. objektu

Struktura komunikačních pararametrů:
(záznam o délce 4 B)

- 2 B Identifikátor PDO
- 1 B Typ přenosu
- 1 B Dodatečný kód

Postup parametrizace modulů C167 pomocí souboru param.h

Obsah hlavičkového souboru *param.h*:

```
#define MODULE_ID 20          /* Cislo modulu (1 - 127) */

#define MASTER                /* Zarizeni je Master */

      /* identifikatory jedn. PDO: */
      //    id = 0 - dany PDO neni pouzit
      //    id = 384 .. 1023 - pro vseob. pouziti pro PDO
#define ID_RPDO1  500
#define ID_RPDO2  501
#define ID_TPDO1  400
#define ID_TPDO2  401

      /* typ prenosu jedn. PDO: */
      //    tt_SYN      synchronni prenos (synchronizace SYNC objektem)
      //    tt_ASYN_RTR asynchronni prenos vyvolany zadosti jineho uzlu
      //    tt_ASYN_EV  asynchronni prenos vyvolany udalosti
definovanou nize
#define TT_RPDO1  tt_SYN
#define TT_RPDO2  tt_SYN
#define TT_TPDO1  tt_SYN
#define TT_TPDO2  tt_SYN

      /* definice udalosti, je-li typ prenosu tt_ASYN_EV pro vysilici
      PDO nebo tt_ASYN_RTR pro přijimaci PDO, jinak 0 */
      //    preddefinovany kod udalosti vyvolane stiskem klavesy: e_KEY
#define EV_RPDO1  0
#define EV_RPDO2  0
#define EV_TPDO1  0
#define EV_TPDO2  0


      /* Mapovani jedn. PDO: (index+subindex objektu v obj. slovniku) */
      /* Pr.: M0_TPDO1  0x707002 */
      /* 1. objekt 1. vysilaciho PDO se nachazi na indexu 0x7070
         a subindexu 02 */

//***  Prijimaci apl. objekty   ***/
// LED1: 0x080000  // LED8 - LED9 (1B)
// LED2: 0x080100  // LED0 - LED7 (1B)
// LCD1: 0x081000  // znak na 1. radku (1B)
// LCD2: 0x081100  // znak na 2. radku (1B)

      /* 1. prijimaci PDO: */
#define     M0_RPDO1    0x080100
#define     M1_RPDO1    0x080000
#define     M2_RPDO1    0
#define     M3_RPDO1    0
#define     M4_RPDO1    0
#define     M5_RPDO1    0
#define     M6_RPDO1    0
#define     M7_RPDO1    0
      /* 2. prijimaci PDO: */
#define     M0_RPDO2    0x081000
#define     M1_RPDO2    0x081100
#define     M2_RPDO2    0
#define     M3_RPDO2    0
```

```
#define     M4_RPDO2    0
#define     M5_RPDO2    0
#define     M6_RPDO2    0
#define     M7_RPDO2    0

//***  Vysilaci apl. objekty   ***/
// AD0: 0x070001  // potenciometr 0 (2B)
// AD1: 0x070002  // potenciometr 1 (2B)
// AD2: 0x070003  // potenciometr 2 (2B)
// AD3: 0x070004  // potenciometr 3 (2B)
// AD4: 0x070005  // potenciometr 4 (2B)
// KEY: 0x071000  // kod klavesy (1B)

      /* 1. vysilaci PDO: */
#define     M0_TPDO1    0x070003
#define     M1_TPDO1    0x070004
#define     M2_TPDO1    0x071000
#define     M3_TPDO1    0
#define     M4_TPDO1    0
#define     M5_TPDO1    0
#define     M6_TPDO1    0
#define     M7_TPDO1    0
      /* 2. vysilaci PDO: */
#define     M0_TPDO2    0x070001
#define     M1_TPDO2    0x070002
#define     M2_TPDO2    0
#define     M3_TPDO2    0
#define     M4_TPDO2    0
#define     M5_TPDO2    0
#define     M6_TPDO2    0
#define     M7_TPDO2    0
```

Postup při nastavování:

1. Nastavení identifikátoru modulu.
   - celé číslo od 1 do 127

2. Volba Master/Slave.
   - vynecháním definice `MASTER` bude zařízení provozováno jako „Slave“.

3. Nastavení konfiguračních parametrů procesních datových objektů (PDO).
   - Identifikátor rámce příslušného PDO. (0 pro neaktivní PDO nebo 384-1023 pro aktivní PDO).
   - Typ přenosu: přiřazení konstanty symbolickým jménům `TT_xPDOx`.
     Možné hodnoty těchto konstant:
     `tt_SYN` synchronni prenos (synchronizace SYNC objektem)
     `tt_ASYN_RTR` asynchronni prenos vyvolany zadosti jineho uzlu
     tt_ASYN_EV asynchronni prenos vyvolany udalosti
   - Dodatečný kód události: symbolickým jménům `EV_xPDOx` přiřadit buď nulu nebo konstantu e_KEY.

4. Nastavení mapovacích parametrů procesních datových objektů (PDO).
   - Postupné přiřazení hodnot indexu a subindexu aplikačních objektů (tak jak jsou uvedeny v komentářích) příslušným položkám mapovacích parametrům. Přiřazování je nutné provádět postupně od položky se symbolickým jménem

`M0_xPDOx`. Je nutné dbát na to, aby celková délka namapovaných apl. objektů nepřekračovala maximální povolenou délku PDO 8 bajtů.

Příklady nastavení komunikačních parametrů:

1. Nastavení 2. vysílacího PDO. Identifikátor 800, typ přenosu synchronní.

```
#define ID_TPDO2  800
#define TT_TPDO2  tt_SYN
#define EV_TPDO2  0
```

2. Nastavení 1. vysílacího PDO. Identifikátor 500, přenos vyvolaný stiskem klávesy.

```
#define ID_TPDO1  500
#define TT_TPDO1  tt_ASYN_EV
#define EV_TPDO1  e_KEY
```

3. Nastavení 1. vysílacího PDO. Identifikátor 450, vysílání jako odpověď na žádost jiného uzlu.

```
#define ID_TPDO1  450
#define TT_TPDO1  tt_ASYN_RTR
#define EV_TPDO1  0
```

4. Nastavení 1. přijímacího PDO. Identifikátor 900, příjem dat bez vyslání žádosti o data.

```
#define ID_RPDO1  900
#define TT_RPDO1  tt_SYN
#define EV_RPDO1  0
```

5. Nastavení 2. přijímacího PDO. Identifikátor 450, vyslání žádosti o data při stisku klávesy.

```
#define ID_RPDO2  450
#define TT_RPDO2  tt_ASYN_RTR
#define EV_RPDO2  e_KEY
```